Transfer learning for crowd numerosity estimation

Pavel Ianko
pavel.ianko@studenti.unipd.it

Abstract

This study covers transfer learning approaches for convolutional neural networks (CNNs), applied to crowd detection task. Using CNNs to assess crowd numerosity is a novel method, compared to computer vision techniques, based on local feature extraction and regression models. For regression algorithms, the pipeline consists of multiple components, dividing responsibility for feature generation and prediction. Instead, a CNN represents a holistic, end-to-end model, encompassing both feature extractor and predictor. In addition, transfer learning enables CNN to use prior knowledge, encoded after pre - training on larger datasets. As a result, CNN with transfer learning techniques has let us outperform the regression-based approach, with a 1.64 mean absolute error on the test set, compared to 3.15 for regression-based approach.

1. Introduction

This work studies Convolutional Neural Networks (CNNs), applied to crowd-counting. Such systems serve the purpose of population density estimation and are useful for measuring the impact of political campaigns, preventing the virus's spread, analysing crowd behavior, or ensuring drone landing safety.

We compare the capabilities of six State Of The Art (SOTA) architectures to predict the crowd numerosity in a shopping mall. To compensate for low size of the dataset (2,000 images), we take advantage of the transfer learning techniques.

At the baseline level, models achieve roughly 2.3 validation mean absolute error (MAE) (fig. 4), with the fixed feature extractor, pre-trained on ImageNet dataset. Based on the number of trainable parameters, training time, and learning curve assessment, Xception is selected as the preferred model. Unfreezing optimal number of top layers, using ELU activation with the correct transfer learning procedure [1] enables us to achieve 1.63 and 1.64 MAE errors on validation and test sets respectively. These results outperform the regression-based and object detection approaches, with 3.15 and 12 MAE errors (table 2).

2. Related work

There is a variety of approaches to a crowd counting task. For example, a target dataset is first presented in the paper of Chen et al. [2]. The authors used a method, similar to the spatial pyramids algorithm. An image was split into cells for computing local visual features. As an image representation, the researchers used a cell-ordered feature vector, which was an input for a multi-output ridge regression model. This approach yields 3.15 mean absolute error (MAE). The combination of feature extraction and regression models is a popular technique, used on other pedestrians datasets [3, 4].

Another common approach is a detection-based system. In the work of Li et al. [5], an adaptive boosting algorithm is used to detect head-shoulder visual patterns. Other authors resort to more sophisticated models, as Bayesian marked point process [6] or Markov chain Monte Carlo methods [7].

Most of the papers, studying the regression-based, clustering-based, and detection-based approaches, are dated back to 2000s, before the onset of deep learning. Today, there is a rising interest in applying CNNs to crowd counting, for drone flight safety [8] and crowd analysis [9].

Based on our results, the CNN capacity for pedestrians detection is much higher, in terms of validation mean absolute error. However, the performance of deep learning models was not actively studied in the research field. For instance, Tzelepi et al. [8] use the fully convolutional neural network of only six layers in depth. We believe, it is reasonable to test deeper architectures and their ability for crowd estimation.

The difference in our approach is an "end-to-end" learning [10]. An end-to-end learning is responsible for both feature engineering and performing inference, embodied in a single model. Thus, instead of having a pipeline, where feature engineering and the model are separate, we will combine them in a deep learning architecture.

3. Dataset

A dataset was compiled by Chen et al. [2] and consists of 2,000 consequential images of pedestrians in the mall,

taken with a fixed camera. Each image has a resolution of 480×640 pixels (height and width respectively), and depicts pedestrians, walking in a crowded mall (fig. 1).

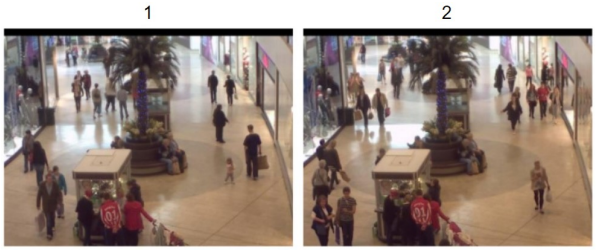

Figure 1. Consequential dataset images

The dataset was randomly split into trainval and a test sets, with a fixed random seed. It was visually assessed, that both sets come from the same distribution. Thus, figure 2 shows, that trainval and test sets are equally representative of any crowd size – ranging roughly from 20 to 50 people, with a mean of around 30. At this step, the test / trainval split corresponds to 20% and 80% (400 and 1600 images respectively). During the training, the trainval dataset is split into train and validation images, in proportions 80% / 20% (1280 and 320 images respectively).

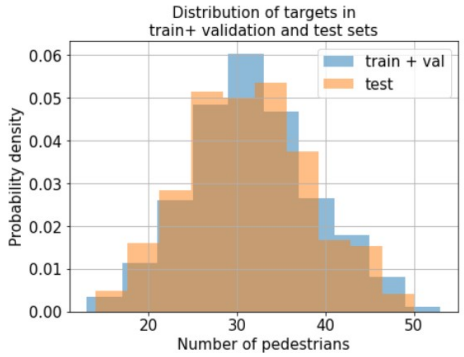

Figure 2. Distribution of target variable (number of pedestrians) in the trainval and test sets

4. Method

This paper concentrates on transfer learning and deep architectures. The work starts with the baseline choice, by shortlisting SOTA models (ResNet50, Xception, Inception v3, Inception ResNet V2, VGG16, and VGG19).

After selecting a promising baseline, we follow A. Geron [1] in our study of the generalization ability, depending on the number of unfrozen layers. This enables us to take advantage of the fine-tuned weights, and save computational resources by fixing the reused layers.

Furthermore, we pre-train the models on PRW dataset [11], originally labelled for object detection. In the final parts, we follow a correct pre-training procedure, described in the book "Hands-on Machine Learning Guide" [1].

	CNNs	Local feature approaches
Pipeline complexity	One model	Multiple components (segment-based, structural, local texture features [2])
Feature generation	Automatic	Manual (depending on pipeline components)
Best MAE (ours)	1.64	3.15 [2]

Table 1. Strengths of convolutional neural networks, compared to other approaches

As a target metrics, we select the mean absolute error (MAE), as it is human interpretable, and allows for comparison with some of the publications and kaggle contributors.

Our approach is meaningful, because CNNs have firmly proved their effectiveness for recognition of visual patterns. As follows from the comparison with regression-based approach [2] and object detectors [12], neural networks outperform methods, based on local visual features. Moreover, CNNs are capable of automatic feature extraction, while preceding approaches require separate pre-processing steps (local texture features, background subtraction, etc.) (table 1). Also, reusing the fine-tuned weights reduces the necessary dataset size, saves computational and temporal resources. As cited from [1], *...it is much more common to reuse parts of a pretrained state-of-the-art network that performs a similar task. Training will be a lot faster and require much less data....*

5. Experiments

5.1. Baseline shortlisting

To select a baseline, we train and evaluate six SOTA architectures - resnet50, inceptionv3, xception, inception resnetv2, vgg16, and vgg 19. Each model was pretrained on the ImageNet dataset of 1.2 million images [13].

For all the models, their feature extractor was appended with the classification block (fig. 3), consisting of a sequence of dense layers with ReLU activation, and dropout layers. Dropout prevents the models from overfitting, while ReLU is a commonly accepted baseline activation function.

The weights of the classification block were initialized randomly. Therefore, for every considered model, seven top layers were unfrozen for fine-tuning, including the top two layers of the feature extractor. This number of layers was already sufficient for adequate validation performance.

The training pipeline consists of fitting the models for at most 30 epochs, with Adam optimizer, batch size of 64

images, and mean squared error loss. As a human-interpretable metrics, MAE error is visually examined. To prevent overfitting, in addition to dropout layers, we add Early-Stopping mechanism, tracking increasing of the validation MSE loss with a patience of 4 epochs. To fine-tune the models on the plateaus, we use the ReduceLROnPlateau callback, provided in keras library.

Notice that each model requires its unique input image size and preprocessing procedure (e.g. resnet 50 requires a BGR color scheme and color centering, while inception v3 necessitates a minmax scaling to the range $[-1, 1]$). We account for these differences in the data preprocessing step.

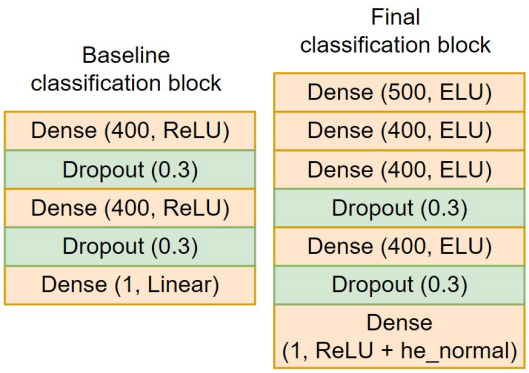

Figure 3. Classification block, that was appended to each of the feature extractors (left); Classification block for the final model (right)

To have an informed decision about the best baseline, we compare the models according to validation performance, number of trainable parameters, training time, and visual assessment of learning curves. Figure 4 summarizes the validation MAE error. Virtually, all models converge to slightly over 2.2 validation MAE, except for VGG16 architecture.

However, as the figure 5 shows, the models perform equally, though vary remarkably in the number of trainable parameters. For instance, inception resnet v2 model achieves roughly 2.2 MAE, using twice as much more trainable parameters.

The plot on the figure 6 visualizes the validation MAE loss for selected architectures. Learning curves suggest several important conclusions. First, the roughness of the learning curves hints on decreasing learning rate and batch size. This was acknowledged in the subsequent part of the paper. Second, we notice the smoothness and stability of learning curve for Xception model. Third - that the steepness of the loss drop for ResNet and VGG16 models is considerably higher, from which follows, that these models can learn faster.

The figure 7 visualizes the time, required for accomplishing one training epoch. From this plot one can conclude, that for the inception resnet v2 architecture, high

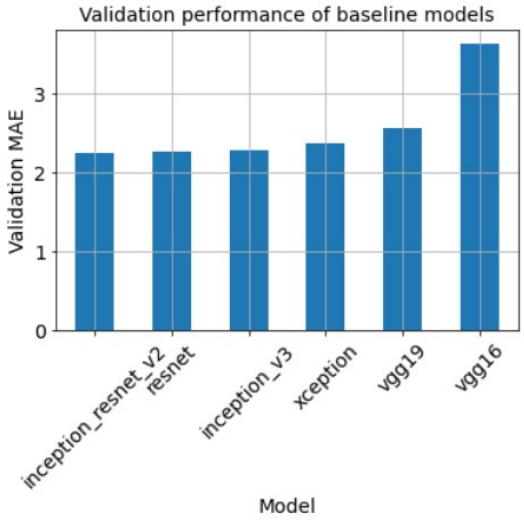

Figure 4. Validation MAE error of considered baseline models

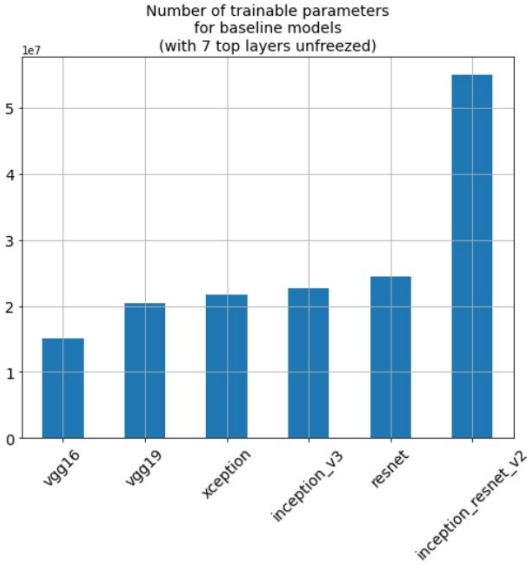

Figure 5. Number of trainable parameters for considered baseline models. Notice, that the frozen part is not considered trainable

number of trainable parameters results in high time demands.

After comparing the baseline models, we select Xception architecture for further experiments. This model has moderate number of training parameters (fig. 5), 2.38 validation MAE error (fig. 4), and is characterized by a smooth loss curve (fig. 6) and an adequate training time (fig. 7).

5.2. Generalization ability VS number of unfrozen layers

Now let us focus on finding the optimal number of unfrozen layers for the Xception architecture. This

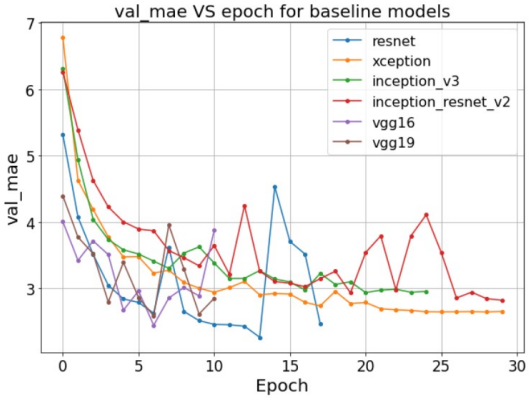

Figure 6. Validation MAE loss curve

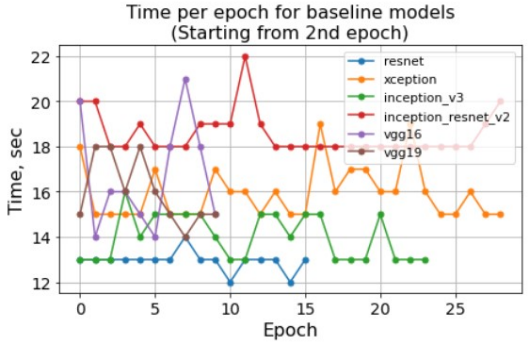

Figure 7. Time, necessary for each model to accomplish one learning epoch

technique was taken from the book of A. Geron [1].

To perform this experiment, we use the same training hyperparameters, as described in the subsection 5.1, changing only the number of unfrozen layers. Figures 8 and 9 summarize the model's generalization improvement, and growth of the number of trainable parameters respectively.

Looking at the barcharts 8 and 9, one can see, how the growth of the parameters quantity corresponds to decreasing the validation MAE error, up to 15 unfrozen layers. Unfreezing more layers results in longer training, and does not improve validation performance. Moreover, the barplot on figure 9 shows a moderate number of trainable parameters for 15 unfrozen layers, yet the model achieves the best validation result (fig. 8). Therefore, we unfreeze the top 15 layers of the Xception architecture during the training. Notice, that this number will change accordingly, in case you decide to deepen the classification block (fig. 3).

5.3. Pre-training with PRW dataset

This chapter is dedicated to a data-driven pre-training technique. Indeed, the 2,000 original images represent only a middle-sized dataset, while high-performance

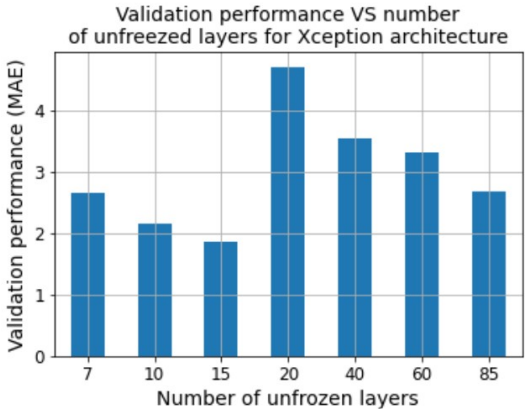

Figure 8. Validation MAE error, depending on the number of unfrozen top layers in Xception architecture

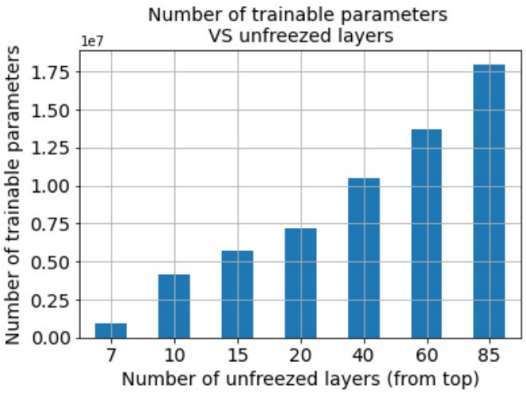

Figure 9. Number of trainable parameters, depending on the unfrozen top layers in Xception architecture

CNNs are trained on tens of thousands of images (e.g. 328,000 in COCO dataset). Therefore, we searched for ways to augment the training data, and give a model higher variety of human postures, appearances, and quantities.

We came across the PRW (Person Re-identification in the Wild) [11] dataset, that consists of 11,816 sequential video frames, captured in six different locations, with a 1920×1080 pixels resolution (fig. 10).

Figure 10. Exemplary series of images from the PRW dataset [11]

The pre-training scheme is composed of the following steps:

- Subsample a pretraining dataset from original 11,816 images;
- Adapt original labels for the regression task (quantity of bounding boxes, instead of their coordinates);
- Reduce PRW images resolution to 360×640 pixels, to have more compatibility with the original dimensions (480×640);
- Reuse the feature extractor for Xception architecture, pretrained on ImageNet dataset, with the appended baseline classification block (fig. 3);
- Pre-train the model for at most 20 epochs, with seven unfreezed layers, batch size of 32 and 20% pre-training data for validation, ReduceLROnPlateau and EarlyStopping callbacks (section 5.1);
- Train the model for at most 30 epochs on the target data, with the batch size of 32 and 20% data for validation, ReduceLROnPlateau and EarlyStopping callbacks (section 5.1).

A figure 11 shows the results of pre-training with PRW images. Pre-training indeed allows us to initiate learning with considerably lower losses (e.g. notice the loss gap between blue and purple curves, corresponding to MAE train errors without pre-training, and with 1637 pre-training images subset).

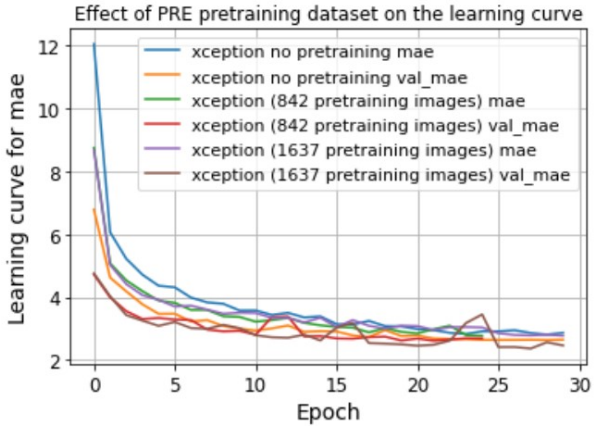

Figure 11. Effect of pre-training with PRW images [11] on the Xception loss curve

Nevertheless, the loss curves converge to roughly equal values, which means, that adding data does not improve much the generalization ability. A slight improvement of validation MAE, from 2.7 to 2.5 is noticed on the first three bars on the figure. 13. However, it is rather attributed to luck, and is not repeatable. Let us share several explanations for that.

First, the PRW data is used at the pre-training step. Consequently, fitting the model on the target dataset overrides the weights, fine-tuned to PRW images. In simple words, the model rewires the knowledge, collected from PRW data.

Second, the original dataset comes from another distribution of person's quantity (fig. 12). Also, taking into account, that CNNs are not scale invariant, subtle differences in persons magnitude, dressing styles (summer clothes on PRW data, and warmer outfit on the target data) might have made it harder for the model to apply the knowledge to the target data.

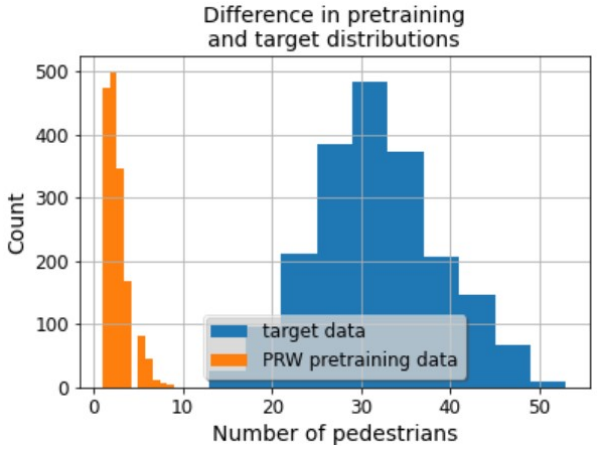

Figure 12. Difference between distribution of target variable in the pre-training PRW dataset [11] and the target dataset

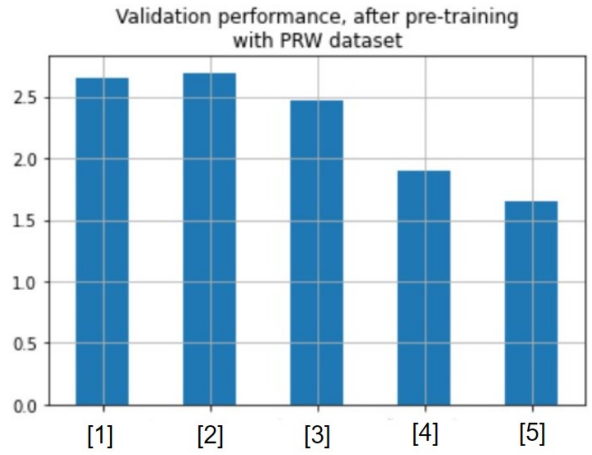

Figure 13. Effect of pre-training with PRW images [11] on the Xception loss curve. [1] - without pretraining, [2] - 842 pre-training images, [3] - 1637 pre-training images, [4] - Geron's pre-training procedure, without PRW pre-training, [5] - best model, with ELU activation and Geron's pre-training procedure, without PRW pre-training

5.4. Correct pre-training procedure

In this section, we follow the pre-training considerations, described in a book "Hands-on Machine Learning Guide" [1]. The roughness of the learning curves (fig. 6) can be explained by the large gradient updates for the classification block (fig. 3) in the beginning of the training. These updates, when back-propagated to the reused layers, can erase the fine-tuned knowledge, updating the weights in an unfavorable direction.

To prevent this, A. Geron suggests the correct pretraining algorithm 1.

Algorithm 1 A. Geron's pre-training algorithm [1]

- Unfreeze only the layers with randomly initialized weights
- Compile the model
- Train the model for 5-6 epochs, giving the random weights meaningful initial values
- Unfreeze all layers, that have to be reused. The number of layers is found experimentally
- Consider decreasing learning rate
- Recompile the model
- Initiate the usual training pipeline

On the figure 13 you can see, that the fourth bar from the left corresponds to the correct pre-training procedure. It has considerably reduced the validation MAE from 2.6 to 1.9, even without pre-training on the PRW dataset.

5.5. Best model and test report

One of the key model hyperparameters is an activation function. In all previous experiments, we used the ReLU activation, which is a decent choice for a baseline model, because of its simplicity, and inference quickness. However, based on the A. Geron's book [1], we conclude, that ELU (exponential linear unit) is a more powerful function, avoiding dying neurons problem, and remaining differentiable in zero. One of the drawbacks of ELU, however, is its slowness at inference time, because of the exponent in the formula.

As the figure 13 shows, changing ReLU to ELU units enables us to decrease the validation loss to 1.6 MAE.

In summary, according to the validation MAE performance, the best model is an Xception architecture, with a deeper classification block (fig. 3), ELU activation function, and the A. Geron's pretraining procedure. During the initial six epochs, top seven layers are pretrained, then the model is fitted for at most 40 epochs, with 17 layers unfreezed. EarlyStopping and ReduceLROnPlateau callbacks are applied. No PRW pre-training is used.

The final model was assessed on validation and test datasets, with 1.63 and 1.64 MAE errors respectively. This reassures us, that we have not overfitted the validation set. After the test estimation, no modification to the model was applied. The final model is available via the github link [14].

Reference	Approach	MAE error
Ours	Xception + ELU + A. Geron's pretraining	1.64
[2]	Regression + local visual features	3.15
[12]	Object detector	12.5

Table 2. Final results, compared with some of the alternative approaches

The plot on figure 14 demonstrates, that the final solution with 1.64 test MAE gives accurate predictions, which outperforms the regression-based approach, and object-detectors trials on kaggle (table 2).

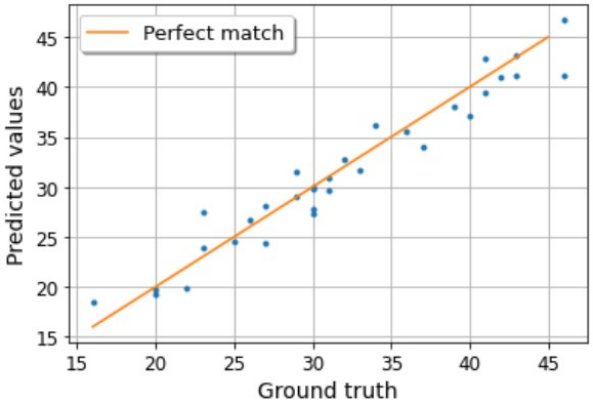

Figure 14. Plot of the best model's predictions on the mini-batch of test images

6. Conclusion

This paper is focused on the transfer learning techniques, to adapt CNNs for crowd counting. The key result of the study is achieving a 1.64 test MAE error, which is almost twice less, compared to regression-based techniques, inspired by local visual features. This shows that an end-to-end learning is a feasible solution for crowd numerosity estimation, and CNNs are capable of extracting meaningful features automatically, without sophisticated pipelines with feature extraction algorithms.

We have learned to use other datasets for pre-training, working with transfer learning keras API, and applying correct training techniques from the book of A. Geron. In the future, it is reasonable to consider more suitable pre-training datasets, and take advantage of the data sequentiality, just like recurrent neural networks account for the sequential relationship between words.

References

[1] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[2] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang. Feature mining for localised crowd counting. In *Bmvc*, volume 1, page 3, 2012.

[3] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–7. IEEE, 2008.

[4] Wenhua Ma, Lei Huang, and Changping Liu. Crowd density analysis using co-occurrence texture features. In *5th International Conference on Computer Sciences and Convergence Information Technology*, pages 170–175. IEEE, 2010.

[5] Min Li, Zhaoxiang Zhang, Kaiqi Huang, and Tieniu Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *2008 19th international conference on pattern recognition*, pages 1–4. IEEE, 2008.

[6] Weina Ge and Robert T. Collins. Marked point processes for crowd counting. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2913–2920, 2009.

[7] Tao Zhao, Ram Nevatia, and Bo Wu. Segmentation and tracking of multiple humans in crowded environments. *IEEE transactions on pattern analysis and machine intelligence*, 30(7):1198–1211, 2008.

[8] Maria Tzelepi and Anastasios Tefas. Graph embedded convolutional neural networks in human crowd detection for drone flight safety. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(2):191–204, 2021.

[9] Lijun Cao, Xu Zhang, Weiqiang Ren, and Kaiqi Huang. Large scale crowd analysis based on convolutional neural network. *Pattern Recognition*, 48(10):3016–3024, 2015. Discriminative Feature Learning from Big Data for Visual Recognition.

[10] Andrew Ng. Machine learning yearning. *URL: http://www. mlyearning. org/(96)*, 139, 2017.

[11] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, and Qi Tian. Person re-identification in the wild. *arXiv preprint arXiv:1604.02531*, 2016.

[12] Ekaterina drasnitsyna. `https://www.kaggle.com/code/ekaterinadranitsyna/crowd-detection-model`.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[14] Demonstration of the final model. `https://github.com/PashaIanko/Kaggle.CrowdCounting/blob/main/2_demo.ipynb`.