CSE 485

Deep Learning

Major Task

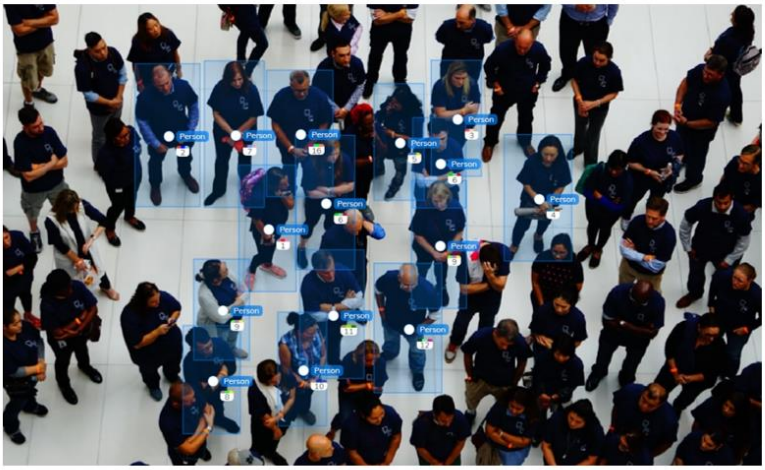
Ahmed Amr Elgayar
19p8349

Table of Contents

Problem statement

55. People Counting

Since the COVID-19 outbreak, people detecting and counting solutions have been growing in popularity, helping to enforce social distancing rules and improve safety.



The goal is to implement the studied deep learning CNNs to solve this problem.

Dataset

The project uses kaggle's crowd counting dataset found in https://www.kaggle.com/datasets/fmena14/crowd-counting/data to compare different pretrained CNN models (resnet50, vgg19, inception v3, mobilenet).

The dataset has 2000 images and their true labels (how many people are in them).

Code

The full code can be found here:
https://github.com/aGayar30/deep_project/blob/main/crowd_counting_comparison.ipynb

After loading and exploring the data, I tried a few data augmentation techniques, but in our case and after several trials, shifting (both height and width), rotations, and zooming resulted in the disappearance of some of the crowd in the photo, resulting in an untrue label which means the loss/error/accuracy will be even lower for all models based on wrong labels. so we decided not to perform these transformations.

Instead horizontal flipping and scaling were the only transformations done before splitting the data set into 80% (1600 images) training and 20% (400 images) Validation(testing).

Then for each model we followed the same pipeline:

i. loading the pre-trained model
ii. Add a global average pooling layer after the last convolutional block.
iii. Removing the last fully connected layer and instead add a (dense, 1024, relu) fully connected layer with only 1 output (the number of people in the image)
iv. Freezing the initial layers (all except last 7 layers) of the pre-trained model (which contain more generic features learned from a large dataset like ImageNet) to retain their pre-learned weights.
v. Unfreezing the later layers (last 7 layers) to adapt the model to the specific features of your dataset by updating their weights during training.
vi. Define the optimizer - this function will iteratively improve parameters in order to minimize the loss.
vii. Set a learning rate annealer - to have a decreasing learning rate during the training to reach efficiently the global minimum of the loss function.
viii. Begin training (only on the last 7 layers)
ix. Plot the loss training and validation curves.
x. Predict on entire validation set.
xi. Plot a scatter plot of True vs. Predicted labels.
xii. Compute MAE (mean average error), MSE (mean squared error), and R^2 score
xiii. Calculate the prediction accuracy, which is the percentage of correct predictions within a predefined threshold (10%)

By comparing all 4 models in the mentioned metrics (MAE, MSE, R^2, Prediction accuracy) we find that the vgg19 model excels above all, the mobile net comes as a close second, the inception is notably worse and resnet50 is the worst.