Project report for DT8807

Jørgen Finsveen

Person anonymization in real-time surveillance cameras

Advanced Topics in Deep Learning with Python

NTNU Ålesund 21.05.2022

NTNU
Kunnskap for en bedre verden

Report

Person anonymization in real-time surveillance cameras

Advanced Topics in Deep Learning with Python

VERSION
2.0

DATE
25.05.2022

AUTHOR
Jørgen Finsveen

NUMBER OF PAGES AND ATTACHMENTS
13 pages and 4 attachments

SUMMARY

Development of deep learning model for surveillance anonymization.

In a world where digitalization introduces surveillance in almost any public places, there is a growing demand of anonymization. Using deep learning, surveillance equipment can by itself censor people on video for their discretion.

This report will revolve around the project of making a deep learning model which can detect and anonymize human faces in real time video. It will describe theory behind DL, discussion around NN architecture chosen for the project, the development of the model and its results and final reflections.

Terminology

Activation function – *Function which transfer the linear combinations of a weighted sum into a nonlinear model*
CNN – *Convolutional Neural Network*
Convolution – *A mathematical operation which takes two functions as input, and returns a third one*
ColorFERET – *A dataset owned by NIST containing thousands of pictures of human faces*
Google Collab – *In-browser IDE which is often used for DL projects*
Keras – An API used for machine learning
Kernel – A matrix representing a convolution filter
Machine learning – *The capability of a machine to imitate human intelligent behavior*
MLP – *Multi-layered perceptron*
Neuron – *A braincell responsible for receiving sensory inputs and returning an output*
NN – *Neural network*
Overfitting – *When the machine learning model is over trained. Leads to the model to memorize the test data rather than learning to recognize it*
Perceptron – *A simple neural network consisting of a single neuron*
Pooling – *Down sampling of an image*
Python – *Programming language often used for machine learning*
R-CNN – *Region-based convolutional neural network*
ReLu – *Rectified Linear Unit, an activation function*
RoI – *Return on Investment*
SoftMax – *An activation function*
SSD – *Single shot detector, a neural network based on CNN*
Supervised learning – *Machine learning through labeled datasets*
TensorFlow – *Open-source library with modules for creating machine learning models*
Underfitting – *When a machine learning model is under trained. Leads to inaccuracy when the model makes predictions*
Unsupervised learning – *Machine learning to analyze and cluster unlabeled datasets*
Weights – *The learnable parameters in the neurons of a network*
YOLO – *You Only Look Once, a neural network based on CNN*

History

VERSION	DATE
1.0	20.05.2022

DESCRIPTION
First draft, missing results, attachments, and reflection

VERSION	DATE
2.0	25.05.2022

DESCRIPTION
Final draft

Table of contents

Introduction

This project was meant for the DT8807 course at NTNU. The purpose of this project is to gain a better insight in deep learning (DL) techniques, and how to approach a machine learning problem. This project has been active for the spring semester of 2022.

Model requirements

Description

The goal of this project is to develop a machine learning model which will be able to detect human faces of people in surveillance cameras, and to censor those in real-time. The developed anonymization model is expected to run on an edge device. The model can be trained using existing datasets or using a newly connected dataset.

Constraints and limitations

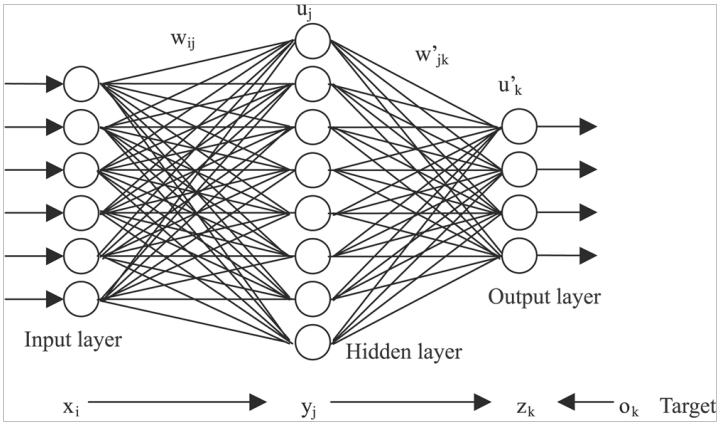
The model is meant to use in surveillance cameras which films crowds of people over time. Therefore, the requirements need to be specified further for certain scenarios.

1. The model should be able to detect and censor multiple faces in a frame, not just one.
2. There needs to be big computational resources required, as it is computationally quite resource-demanding to anonymize the identities of hundreds and thousands of people.
3. The model should be able to identify people from a long range, and even if the persons are blurry or out of focus.

Theory

Visual perception is concept of observing and recognizing objects and patterns trough sorts of visual input as e.g., sight. Vision systems and visual perception is not a new concept, as it is present in many biological creatures. The human vision systems consist of two main elements; the eyes which are the sensing device, and the brain which is the processing device. The ability of perceive something visually and categorize it is a feature which can be imitated algorithmic. This is where machine learning steps in.

At the core of machine learning, there is the neural networks. The neural network is a series of algorithms which mimics the way the human brain operates. Using artificial neurons like these allows us to simulate learning, which opens for a variety of possibilities. Some are image classification, language recognition and picture generating. Machine learning software can make use of neural networks in many ways. In some cases, a machine learning model needs only one neuron, which is known as a perceptron. In other cases, a model requires multiple artificial neurons arranged in many layers, which opens for more advanced learning.



A1: Illustration of a neural network, collected from Craig A. Glastonbury.

The neurons use trial and error to gradually learn something. The weights variables of the neuron are altered until the network is trained. It calculates the weighted sum and makes it prediction by using an activation function. Later, it will compare the prediction to the actual value to calculate an error rate, which again is used to adjust the weights. This pattern is repeats until the error rate is sufficiently low.

A type of neural network which is very relevant for this project is convolutional neural networks (CNN). CNN's are a development of a classic multi-layered perceptron (MLP), that is quite effective to apply to tasks as image classification. The architecture of a CNN consists of an input layer, convolutional layers, fully connected layers, and the output prediction. The convolutional layers are used for feature extraction, where each layer gradually learns more and more complex shapes. The fully connected layers take use of the "knowledge" of the convolutional networks and tries to classify the input image. The prediction is the final output of the network and is based on the classification from the previous connected layers, which are run through an activation function.

Designing a model

Architecture discussion

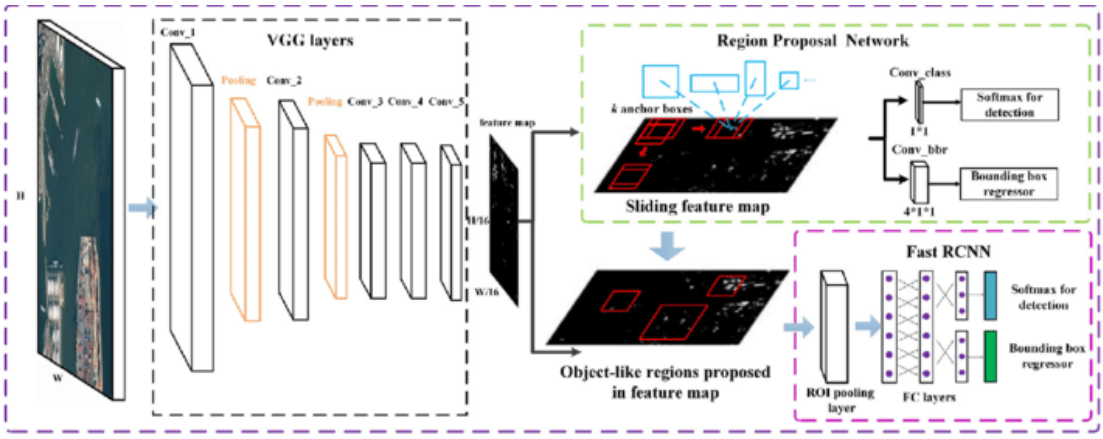
A major issue when constructing a model is to design the architecture of the neural network. For this project, there are several network types that will fit sufficiently. For recognizing faces in frames, will a CNN-based model be a good choice. There are different versions of CNN's with different pros and cons, and the one chosen for this project is a faster R-CNN.

The R-CNN is a region-based convolutional neural network. The R-CNN can be divided into four modules: the selective search region proposal, the feature extractor, the classifier, and the bounding-box regressor. The selective search algorithm is the only module which does not require training. There are several advantages and disadvantages of the R-CNN. Some disadvantages are that the object detection is a slow process and that the training is ineffective in terms of memory usage and time.

The fast R-CNN improves these cons. It consists of four modules as the R-CNN, however, there are some differences. The modules are the feature extractor module, the RoI extractor, the RoI pooling layer, and the two-headed output layer. The fast R-CNN is a lot faster when testing, since

it doesn't need to be fed with thousands of region proposals to the CNN per image. This is being replaced by a convolution operation, which generates a feature-map. There is still one issue. The search algorithm is slow, and it must be generated by another model. This is where the faster R-CNN comes in the picture. The faster R-CNN can be divided into two networks: the region proposal network, and the fast R-CNN. It is in other words an extension of the fast R-CNN.

There are several reasons why this project is based on a faster R-CNN. Its compatibility with our models intention is of course a big factor. It is a solution which reduces the number of models and code-files required to be left with a working product. It is an independent solution and does not require other models to fulfill itself. It is a respected solution which is used in DL-projects of many different degrees of complexity and precision. (*Elgendy: 300*)



A2: The architecture of Faster R-CNN, collected from ResearchGate.

It should also be noted that there are more modern model architectures based on CNN's other than the R-CNN family. Some of these are the You Only Look Once model (YOLO), and the Single Shot Detector (SSD).

Building the architecture

The architecture can be divided into a point-list:

- Importing libraries and modules
- Function for image preprocessing, mapping pixel values, resizing etc.
- Function for extracting CNN features
- Function for extracting box classifier features

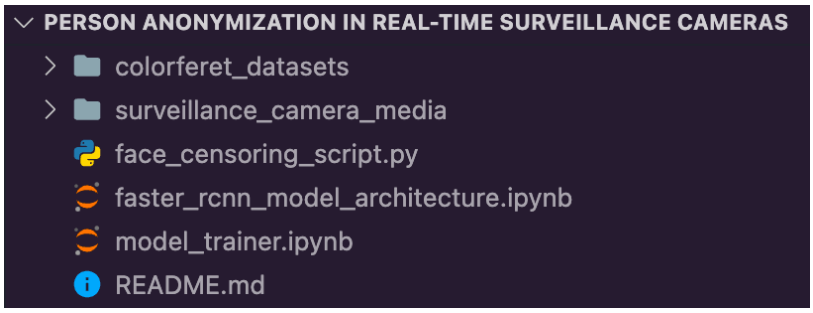
These are the components the model. There are of course additional elements to take into considerations: number of layers, neuron initial weights, kernel configuration and learning rates are examples of these.

Implementation

For my solution, I wish to make a model with a small number of layers. This is because I would like my model training to be fast. For a perfect solution for this project, I would add more layers, which again hopefully will increase the R-CNN's accuracy when identifying human faces.

Furthermore, I intend to have an initial learning rate of around 0,001. This rate may be altered if my models accuracy is insufficient, e.g., if the model gets undertrained or overtrained. The weights need an initial value, but these will adjust by itself in time. The kernel in the earliest layers could be a 3x3 matrix, and gradually expand in the later layers. For activation function, a softmax-function would be appropriate, as this function is being used for multiclass problems with single-label classification.

The faster R-CNN will be coded as a class in its own *.ipynb file. Furthermore, there will be a script for training the model with the desired datasets. The trained model will be stored in a *.h5py-file. The last file required will have functions for reading frames from a surveillance camera as input, apply the trained model, and censor the detected faces.



A3: Initial layout of the program. Screenshot from Visual Studio Code

Resources

Datasets

The model requires to be trained for it to learn the features of the human face. This is also the case for humans, but a computer does not learn as efficiently. It requires thousands and thousands of examples and many training epochs before the algorithm is fitted enough to precisely predict the objects it is meant to identify. Therefore, the model needs to be fed lots of human face images. There are a variety of datasets available to use for facial recognition. Some well-known datasets are the ColorFERET dataset developed by NIST, and the VGGFace dataset by Keras. A big advantage by using datasets like these are that all pictures are of equal resolution and dimensions, so that we do not need to preprocess the images before feeding it to the model. The original idea for this project was to use the ColorFERET datasets to train the model, but in the final solution, the dataset is the VGGFace instead. This is the case since the last one was much easier to obtain.

Libraries

There are a lot of libraries and APIs available for developers to use when creating a machine learning model. For this project, the libraries keras, TensorFlow, sklearn and numpy are some of the libraries used in this project. These libraries contain modules and methods which are to be

used when creating models. Keras, TensorFlow and sklearn are being used for dataset import and model-creation, while numpy are being used for plotting statistics and information in graphs.

Environment

For machine learning, there exist many appropriate IDEs to use. Some are Anaconda, Visual Studio Code and Google Colab. This project has been developed in Google Colab. This environment is remarkably practical for machine learning, as the in-browser IDE has many popular APIs and libraries already installed, and it allows the training to be run on server and not on the local computer. It also offers GPUs which accelerates the processing time by a lot.

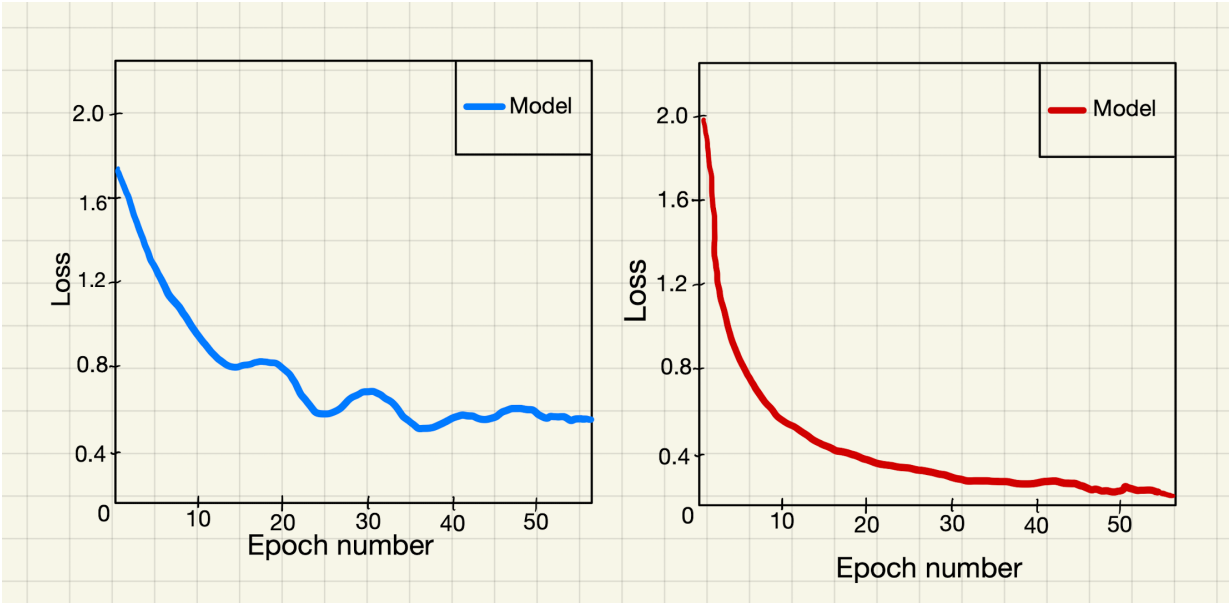
Results

Note that since there were issues preventing the actual training of a model described in this project. This section of the report will therefore be describing different scenarios which are likely to appear, and a description of how to handle such events.

Error rate

A models weights, learning rate and has a big influence on the trained models accuracy. This error-rate, also called loss, tells us how accurate the model was to label test data during each training epoch. We always strive to achieve minimal loss.

It is usual to plot the loss as a function of epochs to see how precise our model gets.



A4: An illustration of a bad loss-development vs. a good loss-development. Hand drawn with GoodNotes.

Over, there are two examples of a models loss-function. Notice that the red function is more consistent and approaches 0. This is a sign of a well-trained model. The blue function is less consistent and ends up with a loss rate closer to 1 than 0. If our model were to have a loss function as the blue one, the first step to improve the accuracy would be to change parameters. There can be many reasons for a model to be inaccurate, but one of the most frequent and the easiest to improve is the learning rate. A too big learning rate leads to overfitting as well as to big adjustments of the neuron weights between each epoch, which again results in inaccuracy and that the model memorizes the dataset rather than learning its features.

Other reasons for big loss values could be that the model architecture is not optimal. This could be minor faults as kernel sizes, padding and activation functions, or it could be more fundamental faults e.g., the model architecture is less effective for this type of deep learning problem.

Usability

A trained model should, by the rule of thumb have an accuracy of at least 70%. A model with an accuracy between 70% and 80% percent is considered as a good model. An accuracy between 80% and 90% would be an excellent model, but an accuracy between 90% and 100% means that your model likely is overfitted, hence, it will not be as accurate when applied to problems which are not part of the dataset it was trained on.

Our model is required to recognize and censor several human faces in a single frame, both faces up close and faces in the distance. This means that our model should be quite precise for it to be capable to handle such tasks. Therefore, the desired accuracy percentage should be closer to 80% than 70%. If this is not achieved, the natural step would be to retrain the model, with new hyperparameters, more layers in the NN, if not with a refined architecture.

If our model reaches an accuracy higher than 75%, it would be considered sufficiently accurate. From this base, the trained model can be applied to censoring script. This script should also be tested on sample footage to make sure that the models performance meets this projects requirement.

Reflection

Issues

Unfortunately, there were several issues appearing during this project. My original plan was to use the ColorFERET dataset to train the model on, but I struggled with importing the dataset. The same thing happened with the VGGFace dataset. I tried many supplements, but both Anaconda, VSC and Google Colab somehow refused to import them. This also happened with some of the libraries from keras that I required. Google Colab threw errors saying that there were no available versions to import. These issues combined with a slight lack of available recourses for this project lead to that our model could not be trained.

Furthermore, I was struggling to find a suitable project, which resulted in reduced time to build the model. The course related to this project is meant for doctorate students, which has lots of more

experience compared to me, who is working on my first year on my bachelor's degree. Picking a project was hard for me, as it would have little relevance to the degree I am working on. Many of the PhD. students were picking projects based on their degrees subject, so I can imagine it was slightly easier to pick a project.

Solution

The issues that I met were unfortunately quite fatal in terms of developing a working project. Therefore, I have prioritized to focus on the theory, and code what I was able to, even though the program could not be run. I have developed parts of the faster R-CNN model, and I have been working on understanding the code and making the code both effective and robust. Even though I may not have a working final product, I am confident that I could make one if I had more time, and if I could solve the problems with Google Colab imports.

Experience

In future projects, I would benefit from the experience made from this project. I will be more capable to locate and choose a fitting machine learning model, and I have better knowledge of setting up a project like this. However, in future projects, I will make sure that I can make better use of the time by having a project idea ready in advance. Also, I would research more thorough for datasets to use, and hopefully be able to resolve importing problems as the ones I have encountered during this project.

Conclusion

There may not be a finished product to show for after this project, but it has regardless been both an interesting and educational project to work on. The main purpose of this project was to gain better insights in deep learning techniques, and how to approach a machine learning problem. In those terms, I would consider the project a success. I am confident that with working equipment, I would be able to create a working solution using the theory and implementation mentioned in this report. Deep learning is a technology which constantly develops, and it is has a great potential. The person anonymization project solves an important privacy problem, and this type of technology will most certainly be a big part of future solutions as well.

Literature

Delua, J. (2021, 03.12). *Supervised vs. Unsupervised Learning: What's the Difference?* IBM. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

Elgendy, M. (2020) *Deep Learning for Vision Systems*, Manning, Shelter Island

Sharma, P. (2018, 04.11). *A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes).* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/

The architecture of Faster R-CNN [Image]. (2018) Gathered from: https://www.researchgate.net/figure/The-architecture-of-Faster-R-CNN_fig2_324903264

Xu, Y. (2018, 20.11). *Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images Dataset V4.* Towards Data Science. https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a