Tampere University

Roosa Kuusivaara & Väinö-Waltteri Granat

Image Recognition - Report

Faculty of Information Technology and Communication Sciences (ITC)
Laboratory Report

Abstract

Roosa Kuusivaara & Väinö-Waltteri Granat: Image Recognition - Report
Laboratory Report
Tampere University
Master's Degree Programme in Signal Processing
October 2023

This report documents the work done in the Image Recognition assignment as a part of the Advanced Signal Processing Laboratory course. In the assignment we familiarize ourselves with modern machine learning, in particular deep learning, and apply them to the task of building a smile detector for real-time execution. The goal is to achieve an accuracy of at least 85% in classifying images based on facial expressions, smiles or non-smiles, using GENKI-4k dataset for training the network.

Keywords: Laboratory Report, Machine Learning, Deep Learning, Image Recognition

Contents

1 Introduction

In this report we describe our work done in the 'Image Recognition' laboratory assignment for the Advanced Signal Processing Laboratory. In this assignment we were to implement a system that would detect if a person was smiling or not from a live video feed, using Machine Learning approach, more specifically a Convolution Neural Network trained as a binary classifier.

The system consisted of two major modules. First a neural network which could classify smiling and non-smiling images with a minimum 85% accuracy. The second module would capture live video feed from computers web camera, from which the module would capture a face from each frame. These frames would then be given for the network to classify if that captured face was smiling or not. The classification would then be shown in the programs UI to the user.

1.1 Neural networks

Neural networks are Generally neural networks are trained using the gradien descdent algorithm.

1.2 Face detection

2 Methodology

2.1 Dataset

For this assignment we were required to use the GENKI-4K dataset **genki**. GENKI-4K consists of 4000 images of faces, labeled either smiling or not smiling. This data set was to be randomly split into portions of 80:20 for training dataset and testing dataset.

To be able to input the GENKI-4K images into the neural network we resized the images to match the required 64x64 pixels size used by the network. The images were also normalized to values 0...1. This is generally recommended to prevent issues with division and square root operations that would happend when using discrete integers.

2.2 Base model implementation

The described model was implemented as Pytorch **pytorch** model.

The base model didn't perform as well as was required by the assignment instructions, so we implemented multiple methods that are generally known to improve the accuracy of image classification models.

2.3 Improved models

Since the base model was a relatively small network, we decided to start optimizing accuracy by increasing the number of layers in the network. The basic idea was that by increasing the number of layers the network would be able to learn more detailed information and capture more of the latent features and thus be able to more accurately make predictions. The danger of increasing the size of the network is that each added parameters increases the training time and more importantly increases the prediction time. The increased prediction time could mean that our program would not be able to make predictions of real time video fast enough to be usable.

We ran a test were we trained models of different size with the same hyperparameters, to find what kind of layers would have the most benefit for the accuracy of the predictions. We noticed that by encreasing the the number of larger layers had more of a impact.

2.4 Optimizing Hyperparameters

The next step to increase the performance of the network was to optimize our hyperparameters. This is usually a difficult problem so we focused only on the following parameters: learning rate, number of epoch and batch size. The best choice for the hyperparameters is dependent on the network we decide to use, so we tested the base model and two of the best performing bigger models to find the optimal model and accompanying parameters.

We also experimented with two different optimizers, Adam and AdamW. AdamW uses the same optimization algorithm as Adam, with the addition of dynamic learning rate (TODO:CONFIRM THIS!!!). Dynamic learning rate allows the optimizer to change the learning rate during training. In general we want to start with a high learning rate to find the area of local maximum fast and the use increasingly smaller learning rate to find the lowest loss. This should make the training faster and prediction a bit more accurate.

2.5 Data augmentation

Since the GENKI-4K dataset is a very small dataset in today's standards we used data augmentation to increase the amount of training data available. In the augmented dataset we included all the original images as such, plus 2 augmented images of each original images.

We used 3 different augmentations methods: flipping creates a mirror image relative to the y-axis, rotation rotates the image 90, 180 or 270 degrees, and finally color jitter changes the saturation of the images. The augmentations we applied at random during augmented dataset serialization and one augmented images be applied with 0 to 3 augmentations.

Finally we experimented with grayscale images. Greyscale images consist only of 1 channel pixels, where as color images use 3 channels. This means that neural network that takes only grayscale images has less parameters, and therefore faster predictions when compared to color images. Our hypothesis was that since smile should be classifiable from both grayscale images and color images equally well, the grayscale models might use the freed parameters to make more accurate predictions faster. We also created a grayscale version of the augmented dataset.

3 Results

3.1 Base Model

3.2 Larger Models

To find a better model we experimented with larger models by adding non-downsampling blocks after the each side of downsampling block. The base models structure in our syntax is: 1x64 2x32 2x16 1x8. The last layer of each size is always a downsampling layers and the ones before that are non-downsampling layers. The extra layers should be able to capture more features which should enable for better classification. For this experimment we trained for 150 epochs to ensure that the larger models had enough time to learn.

Model	Accuracy
2x64 3x64 3x16 2x8	0.82
3x64 4x64 4x16 3x8	0.66
4x64 5x64 5x16 4x8	0.63
2x64 2x64 2x16 1x8	0.80
3x64 2x64 2x16 1x8	0.81
1x64 4x64 4x16 1x8	0.83
3x64 3x64 2x16 1x8	0.80

Table 3.1 *Training accuracy and loss for different CNN models with Adam optimizer*

Based on table results shown in table 3.1 we can make an hypothesis that increasing the depth in the beginning of the network, where the layers are larger has greatly more effect to classification results.

3.3 Hyperparameter Optimization

3.3.1 Learning rate

We tested multiple learning rate for both adam and adamW optimizers. These results are shown in the table 3.2. From the results we can see that the use of adamW didn't really benefit us, not in terms of training time or improved accuracy. This experiment helped us to better narrow down the optimal learning rate for our model.

We decided to do another experiment with learning rates closer to the values of 0.0002 which was determined to be best magnitude in the previous experiment. These results are shown in the table 3.3

Learning Rate	Adam	AdamW
0.2	0.62	0.55
0.02	0.56	0.56
0.002	0.58	0.55
0.0002	0.79	0.76
0.00002	0.62	0.60
0.000002	0.56	0.54
0.0000002	0.58	0.52
0.00000002	0.79	0.51

Table 3.2 *Training accuracy for Adam and AdamW optimizers with different learning rates*

Learning Rate	Adam	AdamW
0.00005	0.63	0.61
0.0001	0.66	0.70
0.0003	0.64	0.62
0.0005	0.62	0.62
0.0007	0.61	0.58

Table 3.3 *Training accuracy for Adam and AdamW optimizers with different learning rates*

From the results we can again see that the choice of optimizer doesn't have much effect in our cage, but it's clear that the learning rate of 0.0001 has produced best results for both of the optimizers.

3.3.2 Batch Size

Batch size is an improtant factor to consider during the training. Using a large enough batch size will allow the model to be trained faster than with a smaller batch size. Large batch size might also faciliate for some randomness in the optimization preventing the model from getting stuck in local minimas. We tested some possible batch sizes as shown in the table 3.4.

Batch Size	Accuracy
1	0.80
8	0.59
16	0.60
32	0.56
64	0.56
128	0.54
256	0.55

Table 3.4 *Training accuracy for Adam optimizer with different batch sizes*

From the results we can see that the batch size didn't have a massive effect on the accuracy. This is most likely since we train for so many epochs. There is a clear outlier in batch size of 1. This might be due to a lucky change in inital weights, but generally larger batch size a used with image classification.

3.4 Data augmentation

We tested the following model: 1x64 3x32 3x16 1x8 with all the datasets described in 2.5. These results are shown in table 3.5.

Dataset	**Accuracy**
Base	0.85
Grayscale	0.75
Augmented	0.87
Augmented Grayscale	0.83

Table 3.5 *Training accuracy and loss for Adam optimizer with the specified datasets*

From the results we can conclude that our hypothesis about grayscale leaving more space for parameters to find features was not correct, since the grayscale datasets performed clearly worse then the color sets. We can also see that data augmentaion improved our accuracy a bit.

3.5 The Final Model

After all the experimentation we settled on the following model for our training:

Since we were limited on time and processing hardware we cannot be sure that this given model is the most performant model that we could have achieved. In our testing the model achieved a accuracy of: xx% and was sufficiently accurate and fast when we tested it in the completed system.

4 Conclusions