# SMILE RECOGNITION

*Jesper Granat, Peter Todorov*

jesper.granat@tuni.fi, peter.todorov@tuni.fi

## ABSTRACT

In this study, a convolutional neural network was trained to recognize smile from an image. The training and deployment was done using the Keras framework on top of the Tensorflow framework. The training data was 4 000 images of smiling and non-smiling faces. The accuracy on this dataset was 86,XXX %.

## 1. INTRODUCTION

Since their introduction [REFERENCE], Convolutional Neural Networks (CNNs) have been widely used in solving various image processing problems, such as classification, object recognition and image enhancement [REFERENCE]. In this study, a custom design CNN is trained for real-time smile recognition. Real-time in this context refers to inference speed where the network can be run on live video feed in over 20 fps on a laptop with GPU.

## 2. BACKGROUND

Convolutional Neural Networks (CNNs) have been shown to be essential in modern computer vision applications, such as classification [REFERENCE]. Classification in particular is rather difficult problem due to its complexity. Here, the goal is to assign a label from a predefined set of labels to an image. Difficulties in this task include truncation of objects (object is partially not in the image), occlusion of objects (some other object is blocking some parts of the desired object), variations in color and lighting, and intra-class variation. The complexity of classification can be intuitively explained by considering a case where an image of a cat is desired to be classified. The problem here is learning a representation of a cat, as many cats can look different, an image of the same cat from different angle and lighting yields in a completely different matrix of pixel values in the image. [REFERENCE]

Traditional approaches to image classification have utilized hand-crafted algorithms such Histogram of Oriented Gradients (HOG) [REFERENCE]. In such algorithms, pixel color values and pixel proximities are utilized to create a representation of the training data. Since 2012 [REFERENCE], CNNs have been dominant in classification. Particularly, deep CNNs from 2015 [REFERENCE] are still dominant in classification. These can include hundreds or even thousands of layers of convolutions. Typical CNNs for classification include two types of layers: convolutional layers and pooling layers. Pooling layers introduce non-linearity to the model to make is feasible to include several layers of convolutions. Convolutional layers, on the other hand, are responsible for creating the image representation. The last layer(s) in CNNs are usually fully connected layers. The output of the network are the probabilities of each class being in the image.

CNNs for classification are trained with cross-entropy loss [REFERENCE]. The idea of the loss is to punish weights that are responsible for wrong classification, and reward weights that are responsible for correct classification during training. Additionally, batch normalization [REFERENCE] is used to generalaize the results. Furthermore, classification with CNNs includes many more bells and whistles that create the absolute best results.

## 3. SOLUTION

The CNN model used in this study will be trained on GENKI-4K Face dataset [REFERENCE]. In addition to the smile status, the training data contains also the face orientation, but that will be ignored in this study. It could be utilized to orient the faces upright as a preprocessing step, but that is out of scope of this study. Therefore, the problem to be solved in this study is formulated as a binary classification problem. The data is split randomly into 80 % training data and 20 % test data in a stratified manner. This means that both splits contain equal proportions of positive and negative examples.

The CNN archtecture used in this study is similar to the one proposed in the laboratory instructions [REFERENCE], and it is implemented with the Keras [REFERENCE] framework. In addition to the layers proposed by the instructor, a batch normalization layer is added in front of every convolutional layer. A $3 \times 3$ convolutional kernel is used in every convolutional layer. Normally distributed glorot initializer [REFERENCE] is used to initialize the

**Table 1**: The confusion matrix for the optimized default architecture with batch normalization

| | Not smiling | Smiling |
|---|---|---|
| Not smiling | 268 | 100 |
| Smiling | 65 | 367 |

**Table 2**: The confusion matrix for the optimized 4 layer architecture with batch normalization and regularization (accuracy 0.86875 %)

| | Not smiling | Smiling |
|---|---|---|
| Not smiling | 309 | 59 |
| Smiling | 46 | 386 |

kernel weights, and each layer contains $L2$-regularization. The activation function for the fully connected layer is Sigmoid activation [REFERENCE]. The model is trained for 50 epochs using Adam [REFERENCE] optimizer and batch size of 32. Otherwise the default Keras parameters are used.

Finally, an additional sanity check and demonstrator is constructed by reading a web camera feed in real time. Each frame is sent to the model synchronously, and the output of the model is displayed in the grafical user interface in a human-readable form. This functionality is implemented with the OpenCV computer vision library [REFERENCE].

## 4. RESULTS

The solution described in Section 3 was achieved through iterative optimizion. First, the default parameters were used, which gave a somewhat reasonable result. Batch normalization was then added, because from our previous experience with neural networks we knew it would help. It did improve the results a little bit. The largest difference in this case, however, was achieved by carefully selecting the loss function. Cross-entropy loss was noted to be the superior loss for this case. Finally, when $L2$-regularization was added, the model beat the desired accuracy.

## 5. CONCLUSION

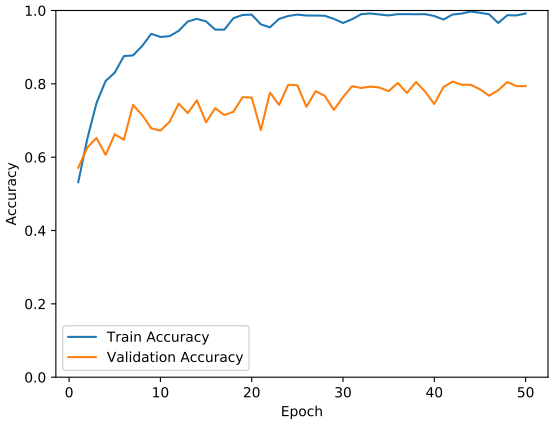


**Fig. 1**: Learning curve with optimized hyperparameters and batch normalization with the default architecture