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 85.36 %.

1. INTRODUCTION

Since their breakthrough in the 2012 ImageNet competition [1], Convolutional Neural Networks (CNNs) have been widely used in solving various image processing problems, such as classification [1], object recognition [2] and image enhancement [3]. 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. 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. [4]

Traditional approaches to image classification have utilized hand-crafted algorithms such Histogram of Oriented Gradients (HOG) [5]. In such algorithms, pixel color values and pixel proximities are utilized to create a representation of the training data. Since 2012 [1], CNNs have been dominant in classification. Particularly, deep CNNs from 2015 [6] are still dominant in classification or as backbones for more difficult tasks, such as image retrieval [7]. 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 in most cases fully connected layers. The output of the network are the probabilities of each class being in the image. [4]

CNNs for classification are trained with cross-entropy loss [8, Ch. 6]. 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 [9] is used to generalize 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 [10]. 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 architecture used in this study is similar to the one proposed in the laboratory instructions, and it is implemented with the Keras [11] framework. In addition to the layers proposed by the instructor, a batch normalization layer is added in front of every convolutional layer. A 3×3 convolutional kernel is used in every convolutional layer. Normally distributed Glorot initializer [12] is used to initialize the kernel weights, and each layer contains $L2$-regularization [8, Ch. 7]. The activation function for the fully connected layer is Sigmoid activation [8, Ch. 6]. The

Table 1: The confusion matrix for the optimized default architecture with batch normalization (accuracy 80.00 %)

	Not smiling	Smiling
Not smiling	287	81
Smiling	79	353

Table 2: The confusion matrix for the optimized 4 layer architecture with batch normalization and regularization (accuracy 85.36 %)

	Not smiling	Smiling
Not smiling	301	67
Smiling	50	382

model is trained using the Adam optimizer [13].

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 [14]. The project was implemented in Python [15].

4. RESULTS AND DISCUSSION

The solution described in Section 3 was achieved through iterative optimization. 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 by roughly 5 %. 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 and one extra layer of a convolution and max pooling was added, the model exceeded the desired accuracy of 85 %. It should be noted that the best accuracy of those parameters was 86.00 % at 44 epochs, which is slightly higher than the final accuracy of 85.36 % at 50 epochs. Even though randomness is used in the experiments, the random seeds were fixed so that the results are both replicatable and comparable. The batch size was 64 in both cases, and learning rate was set to 0.001.

Table 1 shows the confusion matrix for the optimized default architecture with batch normalization. The learning curve for the same process is shown in Figure 1. Respectively, Table 2 and Figure 2 illustrate the same measurements for the final model. As can be seen from the confusion matrices, both models are quite good at classification, however, the final model results in sginificantly better accuracy. The one extra layer adds more expression power to the model so that it can learn a more complicated representation for the task. On the other hand,

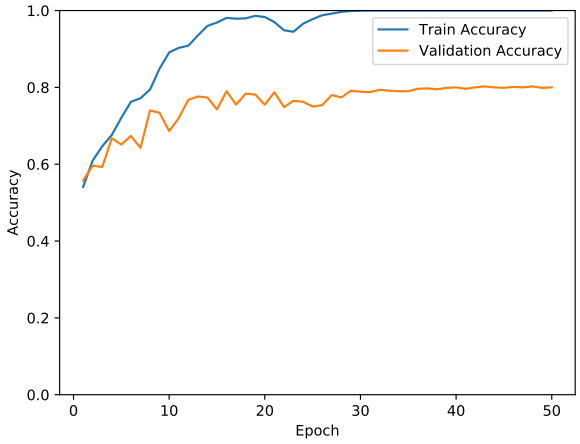


Fig. 1: Learning curve with optimized hyperparameters and batch normalization with the default architecture

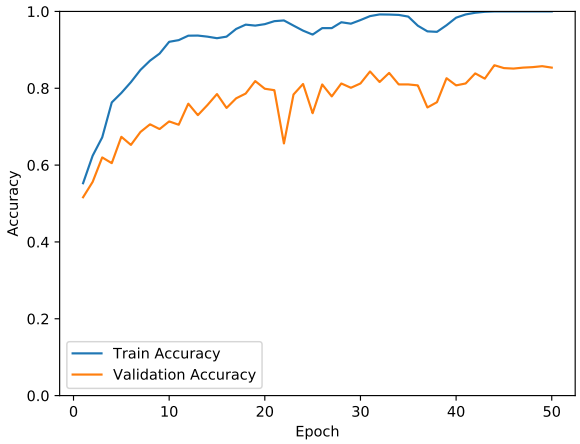


Fig. 2: Learning curve with optimized hyperparameters and batch normalization with one extra layer and $L2$-regularization

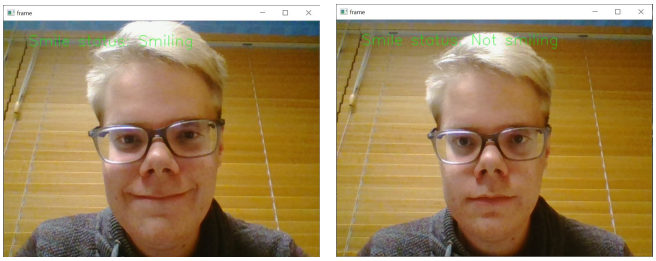
the $L2$-regularization prevents the model from overfitting.

Qualitative analysis shows that the model works somewhat well. Some false positives or false negatives on the test data stem from difficult examples, for example when the person is smirking in the image. This makes it difficult even for humans to classify the images. A true negative and a true positive are shown in Figure 3.

The real-time part of the program works quite well in situations where the person is close enough to the camera. This could be improved in the future by adding a face detector, which would draw a bounding box around the face, after which the input image could be cropped. Also, the lighting plays a part in the working of the real-time functionality. Again, a true negative and a true positive are shown in Figure 4. It is important to set the webcam aspect

(a) True positive (b) True negative

Fig. 3: Qualitative samples from the test data



(a) True positive (b) True negative

Fig. 4: Qualitative samples from the real-time functionality

ratio to one since all training data was square shaped. It is also important to change the input to match the original training input, so here 64×64 is used.

5. CONCLUSION

A model for smile classification was trained in this work. The final model consists of four convolutional, max pooling and batch normalization layers that were initialized with Glorot-initialization algorithm. It was trained with Adam optimizer with learning rate of 0.001 (from the original paper), the loss was cross-entropy loss, and batch size 64. The convolutional kernel size was set to 3×3. It achieves accuracy of 85.37 %.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[3] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. C. Loy, "Edvr: Video restoration with enhanced deformable convolutional networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[4] A. Karpathy, "Convolutional neural networks for visual recognition." [Online]. Available: http://cs231n.github.io/

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[7] M. Teichmann, A. Araujo, M. Zhu, and J. Sim, "Detect-to-retrieve: Efficient regional aggregation for image search," 2018.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[10] http://mplab.ucsd.edu, "The MPLab GENKI Database, GENKI-4K Subset."

[11] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[14] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[15] G. Rossum, "Python reference manual," Amsterdam, The Netherlands, The Netherlands, Tech. Rep., 1995.