Student: Konstantin Kobranov
Program: Data science and economics
Year: 2021/2022
Email: konstantin.kobranov@studenti.unimi.it

Course “Algorithms for massive datasets”

Project “Face/comic recognizer”

1. Introduction

Computer vision is a scientific direction in the field of artificial intelligence and related technologies for obtaining images of real-world objects, processing them, and using the data obtained to solve various kinds of applied problems without the participation (full or partial) of a human. This area has received wide development after the publication of Yann LeCun’s work [1]. In the article, a special type of artificial neural network (ANN), namely convolution neural network (CNN) was applied to handwritten character recognition. Subsequently, CNNs have become widespread in medicine [2, 3], physics [4, 5, 6], and many other fields [7, 8].

One of the basic problems that can be solved using CNNs is image classification. Thus, *the aim of this project* is to develop a classifier that can distinguish between reals and comic faces (see the description of the dataset later).

2. Brief theoretical foundation

The main difference between the regular NNs and CNNs is that the latest take advantage of the fact that the input (usually) consists of images. Hence, the neuron’s input is the three-dimensional matrix. The dimensions are usually called width, height, and depth.

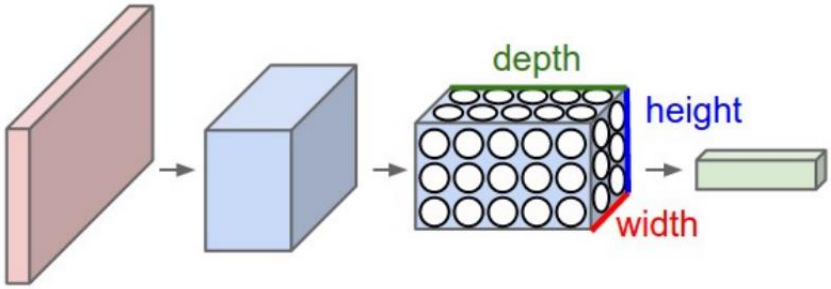

Fig 1 An illustration of the dimension in a CNN [9].

The main layers used to build CNN are the Convolutional layer, Pooling layer, and Fully-connected layer.

Convolutional layer (ConvLayer)

Convolution (consider 1 dimensional case) is an operation over a pair of matrices A (of size $n_x \times n_y$) and B (of size $m_x \times m_y$), which results in a matrix $C = A * B$ of size $(n_x - m_x + 1) \times (n_y - m_y + 1)$, where

$$C_{i,j} = \sum_{u=0}^{m_x-1} \sum_{v=0}^{m_y-1} A_{i+u,j+v} B_{u,v}$$

The convolutional layer applies the convolution operation to the outputs from the previous layer, where the weights of the kernel are the learning parameters.

One of the most important properties of the convolutional layer is local connectivity, i.e., each neuron is connected to the fixed local volume of the input. The size of this region is the hyperparameter called the *receptive field* or the *filter* size (*kernel* size). Additionally, there are three other hyperparameters called the *stride*, the *zero-padding*, and the *depth*. The first one is responsible for the step size with which the filter "moves" during convolution. The second one is used to preserve the spatial size of the input (i.e., pad the matrix with zeros). The last one is the number of filters to be used.

Pooling layer

The pooling layer is designed to reduce the dimension of the image. Hence, it decreases the number of parameters and, as a result, controls overfitting. The hyperparameters are the *size* and the *stride*. One of the most common operations applied in the pooling layer is MAX (*MaxPooling* layer).

Fully-connected layer

The fully-connected layer represents the layer where all neurons are connected to the neurons in the previous layer. The hyperparameter is the number of neurons (units).

3. Methodology

The code for the present project can be found here[1]. The instruction on how to run the code is presented in *README.md.*

3.1.Dataset description

The dataset was obtained from *Kaggle*[2] using *Kaggle API.* It contains 20000 RGB images with sizes 1024x1024. The images are divided into two classes (real and comic faces) each of which contains 10000 images (balanced data).

3.2. Data preparation

Images have been downsized to (96x96) and converted to grayscale. As the result, the input volume has the following dimension (96x96x1). Additionally, all images were normalized, i.e., the value of each pixel was divided by 255. Hence, the pixel values now belong to the range [0, 1]. Of course, it is necessary also to perform mean subtraction. However, below it can be found that the good performance of the model can be achieved without this step.

[1] https://github.com/kkonstantin182/face-comic-recognizer
[2] https://www.kaggle.com/datasets/defileroff/comic-faces-paired-synthetic-v2

Fig 2. An example of the images after processing.

3.3.Model description

The model used in the present work is based on the VGG model architecture, namely VGG-13 [10], with modifications. Hence, in the original model were added the following layers: the batch normalization layer after every convolutional layer; and the dropout layer after every pooling layer (MAX). Moreover, the input size was changed to (96x96x1) and the activation function after the last fully-connected layer was changed to the sigmoid since the binary classification problem is considered.

The *batch normalization* is the technique that was created to solve the problem of the "bad" weights' initialization and, as a result, stabilize training. The *dropout* is a form of regularization (hence, to control overfitting). The idea of the method is to activate a neuron with some probability *p* (a hyperparameter) or make it in-active otherwise. All these methods were used only during the training phase.

As a result, the complete architecture consists of two convolutional layers with the receptive field 3 and the number of filters 64, two convolutional layers with the receptive field 3 and the number of filters 128, two convolutional layers with the receptive field 3 and the number of filters 256, two convolutional layers with the receptive field 3 and the number of filters 512, two fully-connected-layers with 256 units and 1 unit respectively (see the Appendix). The model was created using *TensorFlow* and *Keras* via subclassing approach.

3.4.Experimental setup

Due to the fact that the data is balanced accuracy was chosen as a performance metric. The dataset was divided into 3 subsets: the training, the validation, and the test subset in proportion 0.8/0.1/0.1. The train and validation sets were used for training and hyperparameters optimization. The test set was used for the final evaluation. After the splitting, the subsets consist of approximately the same number of examples belonging to one of the two classes.

The batch size was set to 32. The binary cross entropy was used as the loss function. The Adam algorithm was used as the optimization method with the default values of hyperparameters proposed in the original paper. The initial learning rate was set to 0.001. The initial dropout rate was set to 0.2. The number of epochs was set to 12. The *Google Colab* environment was used to train and evaluate the model with the GPU acceleration.

3.5.Solution scalability

One of this project's main goals is to demonstrate the proposed solution scales up with data size. Hence, the proposed solution takes advantage of *TensorFlow* and *Keras* which enable to optimize data input pipeline as well as support distributed computations.

Following this, the data input pipeline was optimized using *tf.data API*. The *tf.Dataset.cache* and *tf.Dataset.prefetch* methods were used. The first method is design to "keep the images in memory after they're loaded off disk during the first epoch" [11]. This guarantees not to execute some operations (like reading the file) during each epoch but use the cache data. The second method "overlaps data preprocessing and model execution while training" [11]. This ensures that while the model is executing the training step, the data pipeline is reading the data for the next step.

Finally, to implement the distributed training the *tf.distribute.Strategy API*, namely *MirroredStrategy* was used. This technique allows us to perform the model training using the advantages of multi-GPU systems and, as a result, to speed up the computations.

4. Results and discussion

4.1.Baseline model

The baseline model was trained with the following values of hyperparameters: the learning rate equals 10^{-3} and the dropout rate equals $2*10^{-1}$. The corresponding loss and accuracy curves are represented in Fig 3. From the results obtained, the validation curve reaches the value of accuracy approximately equal to 1 (or 0 in terms of the loss) after the 4th epoch, then after the 6th epoch, it starts fluctuating. This can be caused by several reasons, i.e., too high value of the learning rate, a small value of the batch size, or too large number of parameters in the network in comparison with the size of the dataset.

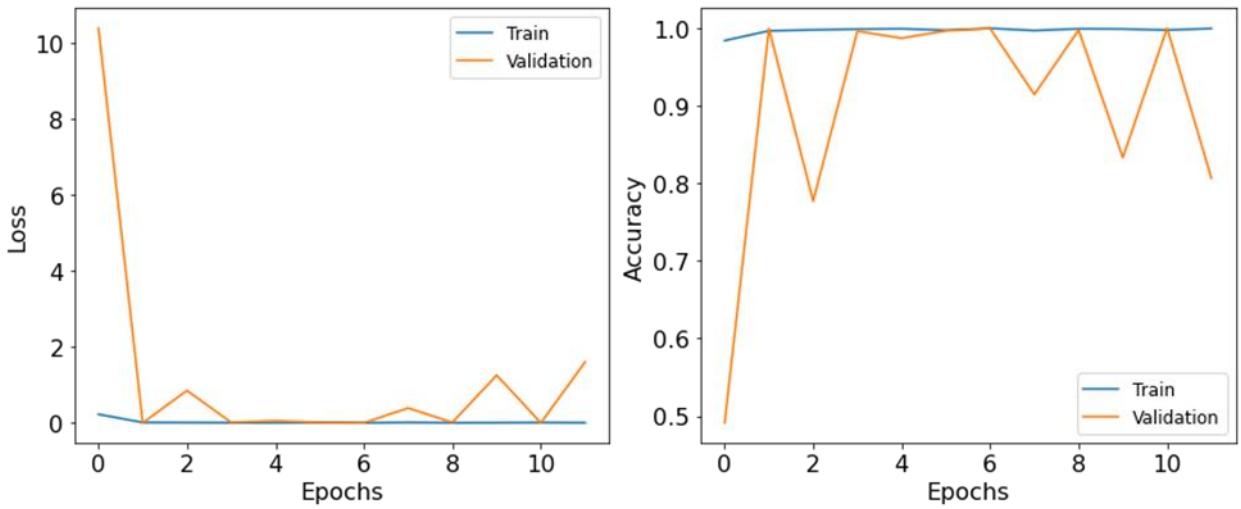

Fig 3. The dependence of the loss (on the left) and the accuracy (on the right) on the number of epochs. The blue curve indicates the evaluation on the training set. The orange curve indicates the evaluation on the validation set. The value of the learning rate is equal to 10^{-3}, the value of the dropout rate is equal to $2*10^{-1}$.

4.2.Optimized model

The two hyperparameters: the learning rate and the dropout rate were investigated. Its values were optimized using the Bayesian optimization. The basic idea of this technique is to use the past best set of hyperparameters to form the new one.

Based on the Bayesian optimization, the following values of hyperparameters should be chosen: the learning rate should be equal to 10^{-4} and the dropout rate should be equal to 10^{-1}. Using these values, the model was trained again. The corresponding results are represented in Fig 4. From the results obtained, no fluctuations are present in this case. Moreover, the overfitting is not observed.

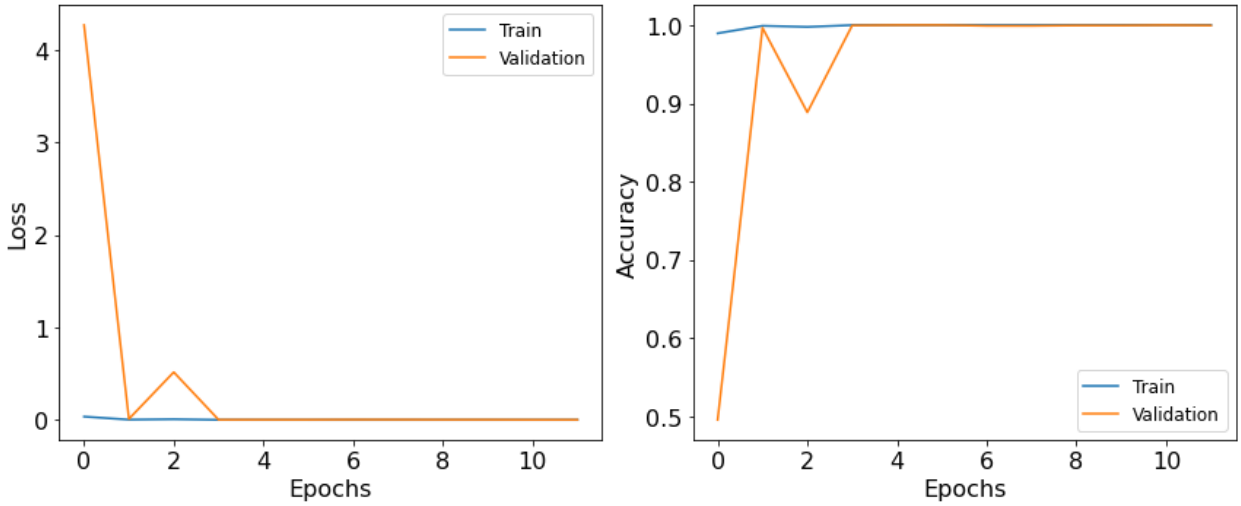

Fig 4. The dependence of the loss (on the left) and the accuracy (on the right) on the number of epochs. The blue curve indicates the evaluation on the training set. The orange curve indicates the evaluation on the validation set. The value of the learning rate is equal to 10^{-4}, the value of the dropout rate is equal to 10^{-1}.

The model with the optimized value of the hyperparameters was used for the final evaluation on the test set. The achieved accuracy is 0.999.

5. Conclusions

As a result of the research done, the CNN based on VGG architecture was developed and improved. The models used in this work have been optimized in terms of the hyperparameters used. The following conclusions have been drawn:

- For the given dataset, the proposed model achieved accuracy on the test equal to 0.999.
- The downsizing and the gray scaling of the input images have not affected significantly the model performance.
- The absence of the mean subtraction has not affected significantly the performance of the model.

6. Declaration

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

7. References

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2323. https://doi.org/10.1109/5.726791

[2] Gavet, Y., & Pinoli, J. (2008). VISUAL PERCEPTION BASED AUTOMATIC RECOGNITION OF CELL MOSAICS IN HUMAN CORNEAL ENDOTHELIUM MICROSCOPY IMAGES. In *Image Anal Stereol* (Vol. 27).

[3] Kayalibay, B., Jensen, G., & van der Smagt, P. (2017). *CNN-based Segmentation of Medical Imaging Data*. http://arxiv.org/abs/1701.03056

[4] Pistellato, M., Bergamasco, F., Torsello, A., Barbariol, F., Yoo, J., Jeong, J. Y., & Benetazzo, A. (2021). A physics-driven CNN model for real-time sea waves 3D reconstruction. *Remote Sensing*, *13*(18). https://doi.org/10.3390/rs13183780

[5] Barry, M. C., Wise, K. E., Kalidindi, S. R., & Kumar, S. (2020). Voxelized Atomic Structure Potentials: Predicting Atomic Forces with the Accuracy of Quantum Mechanics Using Convolutional Neural Networks. *Journal of Physical Chemistry Letters*, *11*(21), 9093–9099. https://doi.org/10.1021/acs.jpclett.0c02271

[6] Cong, I., Choi, S., & Lukin, M. D. (2018). *Quantum Convolutional Neural Networks*. https://doi.org/10.1038/s41567-019-0648-8

[7] Kim, Y. (n.d.). *Convolutional Neural Networks for Sentence Classification*. http://nlp.stanford.edu/sentiment/

[8] Zhou, L., Zhang, C., Liu, F., Qiu, Z., & He, Y. (2019). Application of Deep Learning in Food: A Review. In *Comprehensive Reviews in Food Science and Food Safety* (Vol. 18, Issue 6, pp. 1793–1811). Blackwell Publishing Inc. https://doi.org/10.1111/1541-4337.12492

[9] CS231n: Convolutional Neural Networks for Visual Recognition, accessed 11 December 2022, http://cs231n.stanford.edu/index.html

[10] Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. http://arxiv.org/abs/1409.1556

[11] Load and preprocess images, accessed 11 December 2022, https://www.tensorflow.org/tutorials/load_data/images

Appendix

Input (**96x96x1**)		
1	Convolutional	Filter: (3x3), Num. of filters: 64
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
2	Convolutional	Filter: (3x3), Num. of filters: 64
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
3	Convolutional	Filter: (3x3), Num. of filters: 128
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
4	Convolutional	Filter: (3x3), Num. of filters: 128
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
5	Convolutional	Filter: (3x3), Num. of filters: 256
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
6	Convolutional	Filter: (3x3), Num. of filters: 256
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
7	Convolutional	Filter: (3x3), Num. of filters: 512
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
8	Convolutional	Filter: (3x3), Num. of filters: 512
	Batch normalization	-
	ReLU	-
	MaxPooling	-
	Dropout	-
9	Flatten	
10	Fully-connected	Units: 256
	ReLU	-
	Dropout	-
11	Fully-connected	Units: 1
	Sigmoid	-