Deep learning model for object detection

Cite as: AIP Conference Proceedings **2172**, 020001 (2019); https://doi.org/10.1063/1.5133483
Published Online: 13 November 2019

T. Hristeva, M. Marinova, and V. Lazarov

Deep Learning Model for Object Detection

T. Hristeva[1, a)] , M. Marinova[2,] and V. Lazarov [3]

[1]*Technical University - Sofia, Plovdiv Branch, Plovdiv, Bulgaria*
[2]*Technical University, Sofia, Bulgaria*
[3]*Bulgarian Academy of Sciences, Sofia, Bulgaria*

[a)]thristeva@gmail.com

Abstract. Machine learning is entering in the everyday life of people in different forms. The reasons for this are the continuous development of computer systems, the increase of their computing power and the increase of data stored on electronic media. The main goals of developing self-learning models are to improve or replace existing methods for processing large amounts of information, to improve the services offered by different institutions, and generally to improve and facilitate the lifestyle of modern man. Machine learning can be used to detect complex relationships between a large set of input data, making it an appropriate method for solving a wide range of issues in different spheres such as Bioinformatics, Computer networks, Computer vision, Marketing, Medicine, Natural Language Processing (NLP) and many others.

INTRODUCTION

Artificial intelligence offers opportunities for constructing intelligent systems that can autonomously execute tasks on behalf of the users, and in their benefit. [1]

Machine Learning gives solutions to complex computer problems that are difficult to achieve or even unachievable through the implementation of complex algorithms. There are different definition for machine learning. The most commonly used are:

General definition by Arthur Samuel, 1959

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.[2]

And a more engineering-oriented one by Tom Mitchell, 1997:

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. [2]

Types of ML by training method

Machine self-learning algorithms can be classified by the training method used. The training method is determined by the set of input data to be used.

Supervised learning

In this training, the system receives data that contains input data for the model and labels (desired outputs) for each element of the input data. The model processes the input data, compares the outputs with the desired output, and changes its values (weights) based on the results of the comparison.

A typical task for this type of training is the classification. A good example is the implementation of a spam filter for electronic messages (Fig. 1). The task of the ML model is to distinguish spam from others. Each learning

Proceedings of the 45th International Conference on Application of Mathematics in Engineering and Economics (AMEE'19)
AIP Conf. Proc. 2172, 020001-1–020001-8; https://doi.org/10.1063/1.5133483
Published by AIP Publishing. 978-0-7354-1919-3/$30.00

data message has a label attached to it that indicates whether the message is spam or not. After training, the model is evaluated against validation data and its effectiveness is calculated.

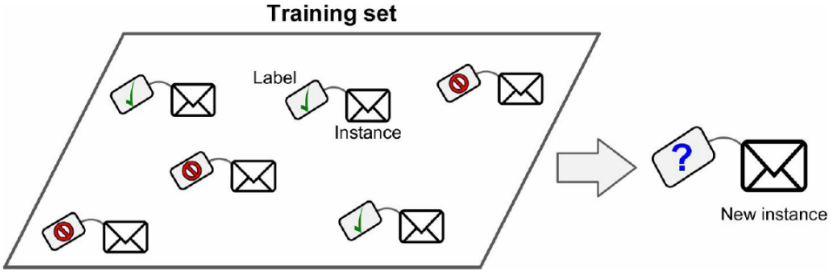

FIGURE 1. Spam filter for electronic messages

Supervised learning algorithms:

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks

Unsupervised learning

In this training, the system receives data that contains input data for the model without their labels. The model tries to find patterns in the input data and groups them against them.

Algorithms for unsupervised training divided into groups according to their purpose:

- Clustering:
  - k-Means
  - Hierarchical Cluster Analysis (HCA)
  - Expectation Maximization

- Visualization and dimensionality reduction:
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning:
  - Apriori
  - Eclat[3]

Semi-supervised learning

In this training, the system receives data that contains data inputs and labels for only part of the input data. First, unsupported training is applied to unlabeled data, followed by supervised training on other data.

The algorithms used in this type of ML are primarily combinations of supervised and unsupervised learning algorithms:

- Deep Belief Networks(DBN)
- Neural Networks
- Clustering Algorithms

Reinforcement learning

In this training, the system (in this type of training it is also called an agent) monitors a given environment and performs a series of actions (called politics) in it. As a result of these actions, the system receives "rewards" and "penalties". On their basis, the model is trained to seek the best policy to get the most wages in a given situation.

Reinforcement learning Algorithms:

- Q-learning
- State-Action-Reward-State-Action(SARSA)
- Deep Deterministic Policy Gradient(DDPG)

Deep Learning

Deep learning is a class of machine self-learning algorithms that is characterized by:

- use hidden layers
- each layer uses the previous one as an input
- training can be supervised, unsupervised or a combination of them
- training by extracting multiple features and dependencies between input data

CONVOLUTIONAL NEURAL NETWORK

Neural networks

Artificial neural networks (or just neural networks) are information processing systems inspired by the biological neural networks that make up the human brain. These systems are taught by task execution without being explicitly programmed with task-specific rules. The main purpose of artificial neural networks is to solve problems that the human brain can solve. Typical examples are computer vision, natural language processing, language translation, and so on.

Biological neural networks are mainly made of neurons interconnected with synapses. Neurons "communicate" with each other by receiving or sending short electrical impulses. Individually, each neuron performs relatively simple actions, but when billions of neurons are networked, they can perform very complex calculations.

Artificial neural networks are constructed in the likeness of biological neural networks. They represent multiple, interconnected units called artificial neurons (a simplified model of biological neurons). Connections (a simplified synapse pattern) between them can transmit messages. These messages are generally real numbers. Each neuron can receive a message, process it and send the result of its work.

Neurons in neural networks are divided into layers. Figure 2 shows a simple neural network. It consists of four layers - input, output, and two hidden layers. The input and output layers do not contain real neurons that perform calculations. They are a graphical representation of the way in which the incoming and outgoing information from the network is connected.[4]

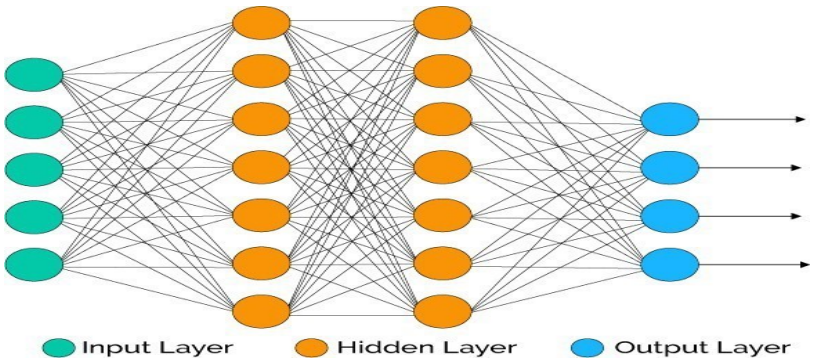

FIGURE 2. Simple neural network

Convolutional neural networks (CNNs)

Convolutional neural networks are part of the "Feedforward" neural networks. They are inspired by how neurons work from the visual cortex of the brain. Each neuron reacts only to a particular region of the field of vision, called a field of perception. Fields of perception of neurons overlap partially and thus cover the entire field of vision.[5]

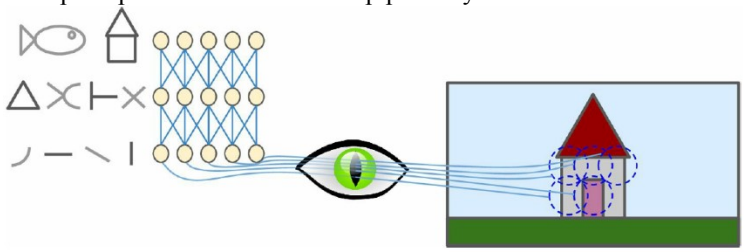

FIGURE 3. The generalized mode of action of the neurons in the visual cortex

Figure 3 shows the generalized mode of action of the neurons in the visual cortex. The first layer neurons recognize only curves. In the second layer, angles and simple geometric shapes are recognized. In the last layer, whole objects are recognized. Convolutional neural networks work in a similar way. They are also made up of a series of layers.

Example Architecture of CNN is shown on Fig. 4.

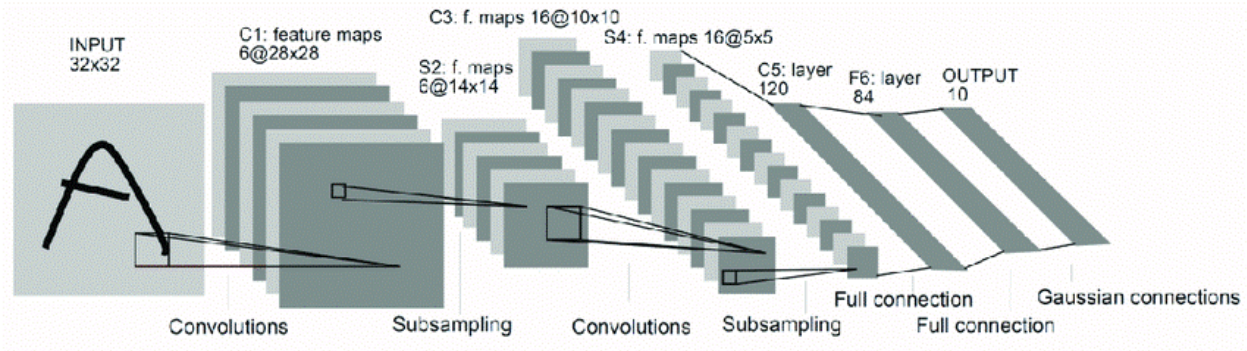

FIGURE 4. Example Architecture of CNN

An important step in CNN development was the development of Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner in 1998. In their article, they present the famous LeNet-5 architecture used to recognize handwritten numbers. Architecture consists of 6 hidden layers. The three main layers are conceivable: the convolutional, the pooling and fully connected layers. The training data used is black and white 32x32 pixels.[6]

Main advantages of CNN

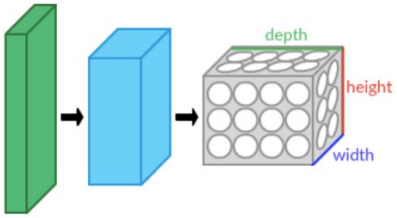

FIGURE 5. 3D volumes with neurons in CNN

- 3D volumes with neurons - neurons are arranged in 3 dimensions: width, height and depth (Fig. 5)
- "Trained" filters in a layer calculate the score only for a given section of the input image without being affected by surrounding plots in it
- Shared weights - each filter applies to the entire visual field. Thus, the position of the object in the image does not matter
- Reusing filters in one layer reduces the number of neurons used in the network

Main layers of CNN

Convolutional layer - this is the main layer of CNN. It contains parameters that vary with each step of learning, called weights and biases. These parameters represent CNN neurons. The formula by which the dimensions of the layer output data are calculated is:

$$N=(W-K+2*P)/(S+1)$$

where

N - width / height output image size
W - Width / Height of the input image
K - size of filters by width / height
P - Padding
S - stride

Kernels - the kernel can be viewed as a window by which the image is scanned with a certain step. The dimensions of the filter are called a field of perception.

Each filter has weights and one deviation. The weights are equal to the product of the filter dimensions, and the deviation is the total for the entire filter. Figure 6 shows the scanning process, which is called a convolution.

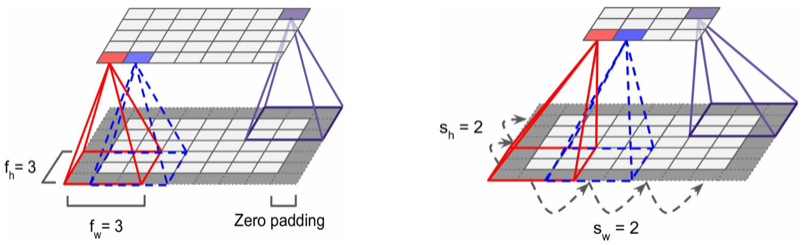

FIGURE 6. A convolution

Rectified Linear Units (ReLU) - this layer applies an activation function to the input data.

Since this layer is used immediately after a convoluted or fully bound layer, it is often not mentioned as a separate layer. There are other activation features that are less commonly used. ReLU is preferred over other features due to its efficiency and easy (fast) calculation. Another activation function used is the sigmoid function.

$$ReLU(x)=max(0,x)$$

Pooling layer - a CNN layer that represents a non-linear reduction of the input matrix dimensions along the x-axis and y-axis. This layer, similar to the conversion layer, "crawls" the data of a given window size (called pool size) through a certain step and performs a certain function of reducing data from the entire window to a single number. Its purpose is to reduce the size of input data before going to the next layer.

Different window reduction features are used, most commonly used by them:

- maximum - finds the maximum of the items in the current window
- average - finds the average of the items in the current window

It is common practice to pool the layer immediately after the conversion layer.

Fully-connected layer - a widely used in all neuron networks layer. With it, each neuron from the current layer has a connection with each neuron from the next layer.

ARCHITECTURE OF THE MODEL

Convolutional layer

This is the main layer in the building of convolutional neural networks. The design of this layer is presented in Fig. 7. It shows the sequence of actions performed by the convolution layer (in the example, this is a layer named conv1). The kernel and bias blocks represent the weights and deviations of the layer respectively. The Conv2D operation performs the kernel-to-input convolution. To the result, add bias - the BiasAdd operation. The activation function - ReLU - is performed on the sum. All the convolutional layers used include this activation function.

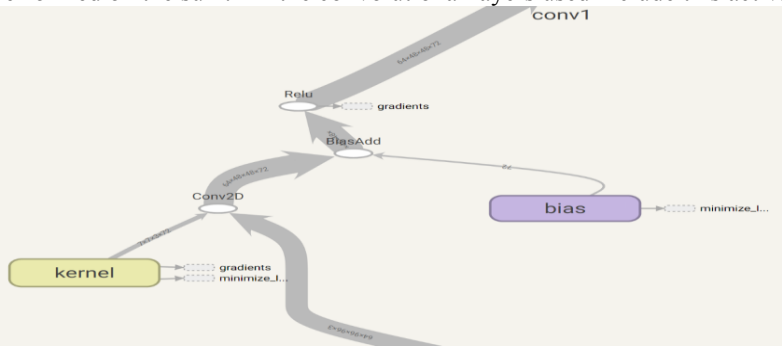

FIGURE 7. The convolutional layer

Pooling layer

Only one action is performed in this layer and it is the application of the MaxPool operation. The layer is used to reduce the size of the input data. In this case, it is reduced 2 times on the "x" axis and 2 times on the "y".

Layer to change the shape of the input data

The created model has only one such layer - flat. Its construction is presented in Fig. 8. It includes only one operation - Reshape. It changes the shape of input data from a 4D tester to a 2D tester.

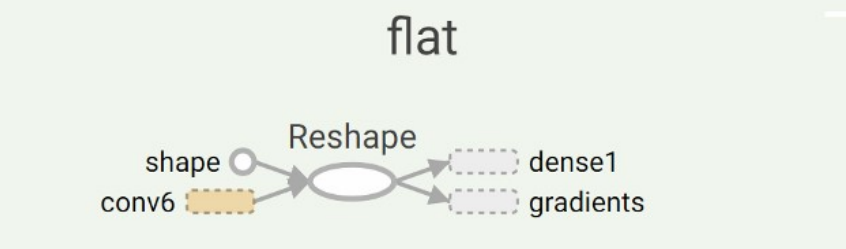

FIGURE 8. Construction of the layer to change the shape of the input data

Fully-connected layer

In the fully-connected the kernel and bias blocks are tensors with weights and layer deviations respectively. The MatMul operation performs a vector work between the kernel weight and the input data. To this result it adds bias - operation BiasAdd. The activation function ReLU is performed on the sum.

The activation function is an addition to this layer. In certain situations, there is no such function. An example of this is the layer of logits. This is the last layer of the model, it calculates the assumptions made by the model. On their basis, the accuracy and probabilities of each assumption are calculated, and the values with which the weights of the model must be changed.

Dropout layer

The created pattern has only one Dropout layer. This layer is used to adjust the input data and is only used in model training mode.

Dropout is a technique to reduce the chances of the model to adapt to training data and to achieve high results only (overfitting). Under this technique, part of the data is removed at random. Sets a ratio of how much of the data will be saved.

Blocks calculating model performance results

- softmax - calculates the probability that an image will be of a particular class for each class
- predictions - makes an assumption for the image class (the one with the highest probability)
- loss - calculates losses from the assumption made. The selected loss calculation function is Softmax Cross Entropy
- accuracy - calculates the accuracy of the assumptions made
- mean - calculate the average of the loss of assumptions for all images
- top_k - finds the two best probabilities in the assumptions made and returns their indexes and labels

RESULTS

Learning rate is important parameter that can be fine-tuned for each model with a specific weight optimizer. The values used for the tests performed for this parameter are: 0.01, 0.005, 0.001, 0.0005 and 0.0001. The weight optimizer used in the tests is AdamOptimizer. The value chosen for the number of dropped elements in the Dropout layer is 0.7 (70%) - the one with the best results from the previous analysis. The number of training steps performed for each item tested is 60000. For each step, 64 images and their labels are submitted.

The learning outcomes of the model are presented in Fig. 9. The y-axis is given the value of the losses from the assumptions made for the labels of the input images. The "x" axis is the number of the steps in which the data was taken.

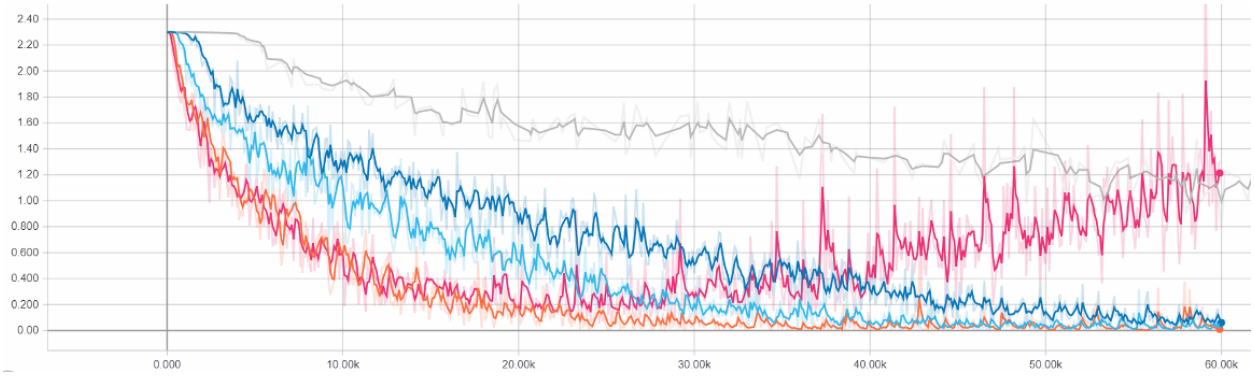

FIGURE 9. Learning results

The chart shows some curious changes. At a training rate of 0.01, the model very quickly reaches low loss rates, but after step 27000, they start to increase, which is not a desired outcome of the training. This change shows that the model learns quickly from the features of the images, but also easily forgets what has been learned so far. For values of 0.005, 0.001 and 0.0005, the model achieves the best results for 60000 training steps. The main difference between them is the speed at which these results are achieved. At the slowest rate of training - 0.0001, the model achieves significant results too slowly. From the diagram, you can see that losses up to step 5000 remain almost constant. The training of the model with this speed was continued to a 300,000 step (of scientific curiosity). Only in steps of 250,000 the model achieves learning losses similar to those at the rates of 0.005, 0.001 and 0.0005 at step 60000.

CONCLUSION

Designed architecture of the model works very well with the classification of images in the selected criteria. The amount and quality of training and evaluation data is the most influential on the results achieved. When data quantity and quality are increased, better results are expected. Despite the limited number of training data, the model achieves approximately 77% accuracy on the data and a loss value of 0.75-0.95.

The techniques used to improve the initial results increase the accuracy of the projected model by 12% to 15%. This is mainly due to the reduction in the effect of overfitting. The greatest benefit to this is the artificial increase of training data, the reduction of the discovered characteristics during the training (through the Dropout layer), the adjustment of the speed of training and the used weight optimizer.

ACKNOWLEDGMENTS

The authors would like to thank the Research and Development Sector at the Technical University of Sofia, Bulgaria for the financial support.

REFERENCES

1. T. Glushkova, M. Miteva, A. Stoyanova-Doycheva, V. Ivanova, S. Stoyanov, “Implementation of a Personal Internet of Thing Tourist Guide”, *American Journal of Computation, Communication and Control*. Vol. 5, No. 2, 2018, pp. 39-51.
2. A. Geron, “Hands-On Machine Learning with Scikit-Learn and TensorFlow”, O'Reilly Media, 2017.
3. A. Coates, H. Lee, A. Y. Ng, “An Analysis of Single Layer Networks in Unsupervised Feature Learning”, AISTATS, 2011
4. A. Krizhevsky, I. Sutskever, G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada.
5. I. Goodfellow, Y. Bengio, A. Courville, “Deep Learning”, MIT Press, 2016
6. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, 1998.