Image Classification Using Convolutional Neural Networks

Madhurima Tummala

Information Technology, VNR Vignana Jyothi Institute of Engineering and Technology, Bachupally, Hyderabad, Telangana, India - 500090

madhu.tummala1998@gmail.com

DOI: 10.29322/IJSRP.9.08.2019.p9261
http://dx.doi.org/10.29322/IJSRP.9.08.2019.p9261

***Abstract*—** Convolutional neural networks (CNNs), consist of multiple processing layers to learn the representations of data with multiple abstract levels. They are the most successful machine learning models in recent years. Within this paper, we will know the usage of a trained deep convolutional neural network model to extract the features of the images, and then classify the images. We will study image processing and understand image classification. It has good application prospects.

***Keywords*—** Convolutional Neural Networks, Deep Learning, Image Pre-processing, Image Classification, Overfitting.

I. INTRODUCTION

The rise of big data and the rapid popularization of high-performance computing devices in recent years have contributed to the unprecedented development of machine learning. Regarding image classification, this study intends to artificially extract features, and convert the features of an original image into useful features characterized by lower dimension and less noise. Machine learning has been gaining momentum over last decades. It is a class of artificial intelligence methods, which allows the computer to operate in a self-learning mode, without being explicitly programmed. Neural network is a machine learning algorithm. It is a system of number of interconnected artificial "neurons" which exchange messages between each other. This network consists of numerous layers of feature-detecting neurons. Each layer consists of several neurons that respond to different combinations of inputs derived from the previous layers. Convolutional Neural Network is a specialized context of the neural network which was proposed by Yann LeCun in 1988. It comprises of one or more convolutional layers, often combined with a subsampling layer, which are succeeded by one or more fully connected layers as in a basic neural network. The design of a CNN is triggered by the discovery of a visual mechanism, called the visual cortex, inside the brain. Each feature of a layer receives numerous inputs from a group of features located in a small neighbourhood which again reside in the previous layers called as local receptive fields. With these local receptive fields, features can extract primitive visual features, such as core, oriented edges, corners, end-points, etc., which are later combined by the higher layers. One of the most important uses of this architecture is Image Classification. In many cases, features derived from the top layer of the CNN are resorted for classification; however, those top layered features may not contain enough or required useful information to predict an image appropriately. In some instances, features acquired from the lower layers carry more distinctive power than those derived from the top. Hence, applying features from a particular layer only for classification seems to be a process which does not utilize deep CNN's potential discrete power to full of its extent. This innate property leads to the demand for fusion of several features from multiple layers.

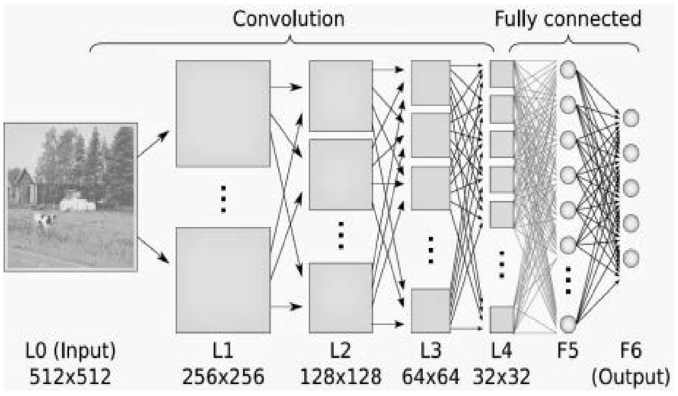


Fig. 1. Convolutional Neural Networks

While neural networks and various other pattern detection methods have been prevalent, there is a significant development in the field of convolutional neural networks. This is because of its ruggedness to shifts and distortions in the image, fewer memory requirements, easier and better training. The image input is passed through a series of convolutional, pooling, flattening and dense fully connected layers, and then the output is generated. In this paper we will be studying about layers of convolutional neural networks and its implementation in real-time.

II. RELATED WORK on IMAGE CLASSIFICATION using CNNs

Convolutional Neural Network has become the most successful deep learning model in the computer vision industry. In recent years, deep learning has been quickly promoted in the machine learning industry, and remarkable theoretical and practical achievements have been continually attained. The development of deep learning can be divided into three stages. From the 1940s to the 1960s, deep learning was called the control theory. While the theoretical development of biological learning and the establishment of the perceptron triggered the first wave of interest in artificial neural networks, as the perceptron could not be used to learn linearly inseparable data, it was resisted by many researchers, which resulted in the low ebb of the first stage. The second wave started with connectionism in 1980 and ended in 1995. In 1974, Paul Werbos proposed the back-propagation algorithm, and solved the problem of linear inseparability in the neural network model. However, as the quantity of training data was small, and the performance of the computers was poor at that time, there was a serious problem in most of the deep learning models - overfitting. At the same time, great progress was made in other machine learning fields. Consequently, deep learning was neglected by most people, which marked the low ebb of the second stage. In 2006, Hinton and Salakhutdinov displayed a neural network model called the autoencoder in the magazine Science. By adopting an unsupervised learning algorithm to initialize networks layer by layer, they efficiently improved the generalization of the deep network. Since then, many researchers have used the same method to revitalize a large number of the deep learning models, which were challenged in the 1980s. This marked the entry into the third wave of deep learning. Based on the visual system, the convolutional neural network (CNN) is a neural network used exclusively to process data with a lattice structure. In 1959, Hubel and Wiesel discovered the visual cortex cells of mammals and proposed the concept of a partial receptive field. Inspired by this, Fukushima and Kunihiko put forward Neocognitron in 1984, which can be regarded as the prototype of the modern convolutional network. In the 1990s, LeCun published relevant papers to define the modern structure of CNN and improved it afterwards. LeNet-5 is the first convolutional network that could be applied in reality. With little pre-processing, the network can directly learn the number image classifications in the original pixels. However, the training data were inadequate, and the computation ability of the computer was weak; thus, LeNet-5 was not effective in handling complicated problems. In 2009, Le integrated the convolutional neural network with the deep belief network to form the Convolutional Deep Belief Network (CDBN). In 2012, Krizhevsky adopted AlexNet to bring the error rate of image classification down from 26.3% to 15.2% in ILSVRC-2012. In 2014, a Google team used GoogLeNet to achieve a lower error rate (6.67%) in ILSVRC-2014. In 2015, the convolutional network of Microsoft MSRA brought the error rate of the ImageNet 2012 data concentration down to 4.94%, thus, surpassing humans in terms of recognition.

III. SYSTEM MODEL

Python Code is used for this project. As a framework we implemented Keras, which is a high-level neural network API written in Python. It can't work by itself, it needs a backend for flow-level operations. Thus, we installed a dedicated software library called Google's TensorFlow. As a development environment we used Jupyter Notebook and Matplotlib for visualization. For network training and testing we can download the dataset of images. This is done by creating an environment using Anaconda software. We can also enhance it by using Object Detection packages supported by python i.e., ImageAI object detection class.

IV. PROBLEM STATEMENT

The human visual system perceives the world in a different manner than digital detectors imposing various additional noise and bandwidth restrictions. Human optical illusions distort and fabricate features due to fundamental properties of human visual perception. Moreover, the memory of humans might me huge but ability to use it might lack. If there are hundreds and thousands of pictures that are to be studied, then it can be digitally possible at ease while it can be a very difficult task manually. The underlying features of the images, distortions, shifts, noise, etc., cannot be easily recognizable by humans. Thus, image classification has been introduced on a digital platform.

V. SOLUTION

Digital image classification utilizes computer-aided enhancement to turn subjective features of an image into data that can be measured, quantified and evaluated. Image processing must be proposed in a manner wherein it must be consistent with the scientific methodologies so that others may use or reproduce, and validate, one's results. To study image processing and understand image classification we will be using convolutional neural networks. The image input is passed as a dataset through a series of convolutional, pooling, flattening and dense fully connected layers, and then the output is generated.

A. Convolution Layer

It is always the first. The image matrix with pixel values is entered into it. Imagine that the reading of the input matrix begins at the top left of image. Next the software selects a smaller matrix there, which is called a filter. Then the filter produces convolution, i.e. moves along the input image. The filter's task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by one unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller than the input matrix. Whenever the image passes through a convolution layer, the output generated by the first layer becomes the input taken by the second layer. And this happens with every further convolutional layer.

$$\mathbf{T}(f(x)\otimes g(x)) = \mathbf{T}\left(\int_{-\infty}^{\infty} f(x)g(u-x)dx\right) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)g(u-x)dx\,\exp(2\pi isu)du$$

Fig. 2. Convolutional Theorem

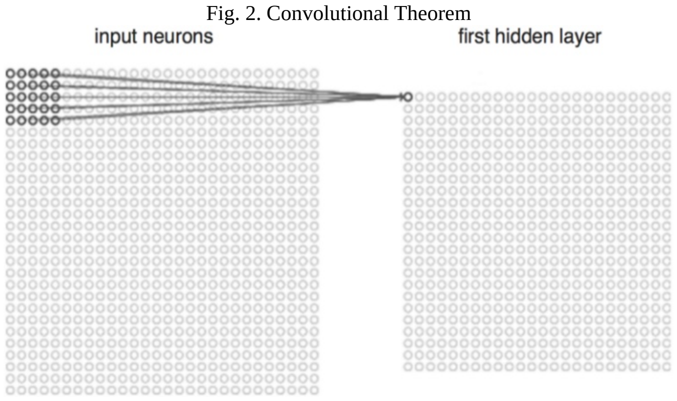


Fig. 3. Convolutional Layer

B. Pooling Layer

It follows the convolutional layer. It works with width and height of the image and performs a down sampling operation on them. As a result, the image volume is reduced. This means that if some features have already been identified in the previous convolution operation, then a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

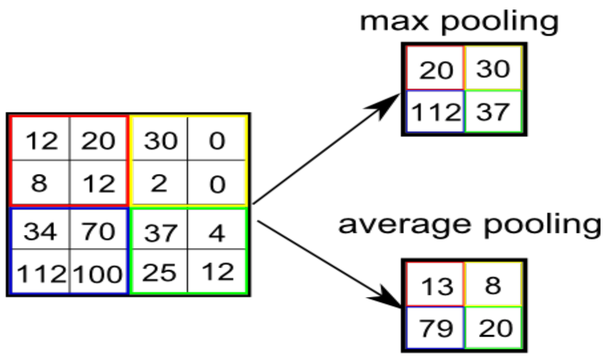


Fig. 4. Pooling Layer

C. Flattening Layer

After the convolution and pooling layers, our classification part consists of dense fully connected layers. However, these fully connected layers can only accept One Dimensional data. To convert our 3D data to 1D, we use the flattening layer.

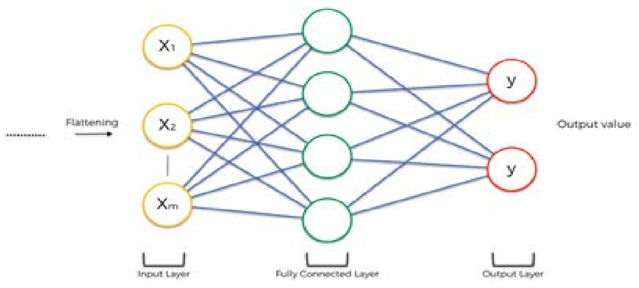


Fig. 5. Flattening Layer

D. Dense Fully Connected Layer

After completion of the mentioned series of layers, it is necessary to attach a dense fully connected layer. By appending a fully connected layer at the end of the network will result in an N-Dimensional Vector, where N represents the number of classes from which the model opts the desired class. Neurons which are in a fully connected layer will have full connections to all the activations occurred in the previous layer.

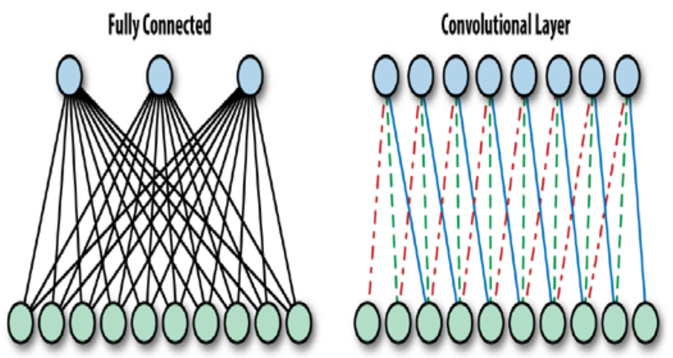
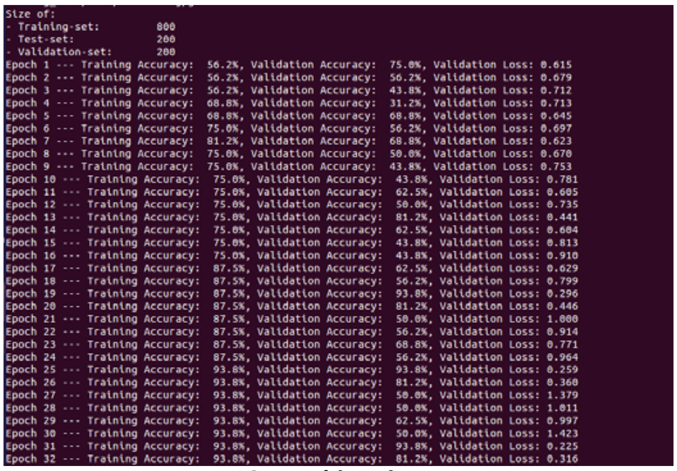


Fig. 6. Dense Fully Connected Layer

VI. ANALYSIS

The basis of the model is Supervised Machine Learning. It is one of the ways of machine learning where the model is trained by input data and expected output data. It is tested and evaluated. To create such model, it is necessary to go through the following phases: model construction, model training, model testing and model evaluation. Model Construction depends on machine learning algorithms. In this project's case, it is convolutional neural networks. Before model training it is important to scale data for the further use. After model construction it is time for Model Training. In this phase, the model is trained using training data and expected output for this data. Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model. Once the model has been trained it is possible to carry out Model Testing. During this phase a second set of data is loaded. This data set has never been seen by the model and therefore it's true accuracy will be verified. Finally, the saved model can be used in the real world. The name of this phase is Model Evaluation. This means that the model can be used to evaluate new data.

The datasets of cats and dogs were considered. The aim was to classify the images of cats and dogs. The Training Set has 800 pictures of cats and dogs. The Test Set has 200 pictures of cats and dogs. Given any image we must classify whether it is a cat or a dog. As a result of testing the model, we got a very good accuracy: 93.8% of correct classification samples after 32 epochs. The only drawback was that we had to wait about 30 minutes until 32 epochs come to the end. If the training data accuracy (acc) keeps improving while the validation data accuracy (val_acc) gets worse, we are likely in an overfitting situation, i.e. your model starts to basically just memorize the data. Consequently, this model is sufficient to train on 10 epochs. The model can still be optimized to get 100% accuracy by introducing new techniques into the model.



```
Size of:
- Training-set:        800
- Test-set:            200
- Validation-set:      200
Epoch 1 --- Training Accuracy:  56.2%, Validation Accuracy:  75.0%, Validation Loss: 0.615
Epoch 2 --- Training Accuracy:  56.2%, Validation Accuracy:  56.2%, Validation Loss: 0.679
Epoch 3 --- Training Accuracy:  56.2%, Validation Accuracy:  43.8%, Validation Loss: 0.712
Epoch 4 --- Training Accuracy:  68.8%, Validation Accuracy:  31.2%, Validation Loss: 0.713
Epoch 5 --- Training Accuracy:  68.8%, Validation Accuracy:  68.8%, Validation Loss: 0.645
Epoch 6 --- Training Accuracy:  75.0%, Validation Accuracy:  56.2%, Validation Loss: 0.697
Epoch 7 --- Training Accuracy:  81.2%, Validation Accuracy:  68.8%, Validation Loss: 0.623
Epoch 8 --- Training Accuracy:  75.0%, Validation Accuracy:  50.0%, Validation Loss: 0.670
Epoch 9 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.753
Epoch 10 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.781
Epoch 11 --- Training Accuracy:  75.0%, Validation Accuracy:  62.5%, Validation Loss: 0.605
Epoch 12 --- Training Accuracy:  75.0%, Validation Accuracy:  50.0%, Validation Loss: 0.735
Epoch 13 --- Training Accuracy:  75.0%, Validation Accuracy:  81.2%, Validation Loss: 0.441
Epoch 14 --- Training Accuracy:  75.0%, Validation Accuracy:  62.5%, Validation Loss: 0.604
Epoch 15 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.813
Epoch 16 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.910
Epoch 17 --- Training Accuracy:  87.5%, Validation Accuracy:  62.5%, Validation Loss: 0.629
Epoch 18 --- Training Accuracy:  87.5%, Validation Accuracy:  56.2%, Validation Loss: 0.799
Epoch 19 --- Training Accuracy:  87.5%, Validation Accuracy:  93.8%, Validation Loss: 0.296
Epoch 20 --- Training Accuracy:  87.5%, Validation Accuracy:  81.2%, Validation Loss: 0.446
Epoch 21 --- Training Accuracy:  87.5%, Validation Accuracy:  50.0%, Validation Loss: 1.000
Epoch 22 --- Training Accuracy:  87.5%, Validation Accuracy:  56.2%, Validation Loss: 0.914
Epoch 23 --- Training Accuracy:  87.5%, Validation Accuracy:  68.8%, Validation Loss: 0.771
Epoch 24 --- Training Accuracy:  87.5%, Validation Accuracy:  56.2%, Validation Loss: 0.964
Epoch 25 --- Training Accuracy:  93.8%, Validation Accuracy:  93.8%, Validation Loss: 0.259
Epoch 26 --- Training Accuracy:  93.8%, Validation Accuracy:  81.2%, Validation Loss: 0.360
Epoch 27 --- Training Accuracy:  93.8%, Validation Accuracy:  50.0%, Validation Loss: 1.379
Epoch 28 --- Training Accuracy:  93.8%, Validation Accuracy:  50.0%, Validation Loss: 1.011
Epoch 29 --- Training Accuracy:  93.8%, Validation Accuracy:  62.5%, Validation Loss: 0.997
Epoch 30 --- Training Accuracy:  93.8%, Validation Accuracy:  50.0%, Validation Loss: 1.423
Epoch 31 --- Training Accuracy:  93.8%, Validation Accuracy:  93.8%, Validation Loss: 0.225
Epoch 32 --- Training Accuracy:  93.8%, Validation Accuracy:  81.2%, Validation Loss: 0.316
```

Fig. 7. Output of the code execution

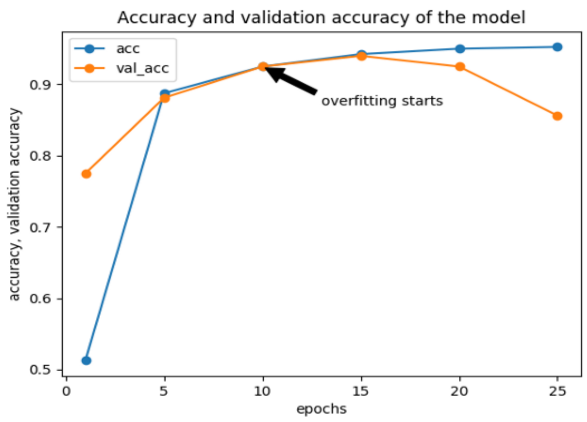


Fig. 8. Overfitting by Plotting

VII. CONCLUSION

From this project, we can figure out what deep learning is. We can construct, assemble, train, test and evaluate the Convolutional Neural Networks Model to classify images of cats and dogs or any image. This model works well with small number of images and we can measure how the accuracy depends on the number of epochs in order to detect potential overfitting problem. It is determined that 10 epochs are enough for a successful training of the model. The scope can be extended to try this model on more data sets and apply it to practical tasks in order to see how a higher efficiency can be achieved in various problems.

ACKNOWLEDGMENT

This project work was supported by Information Technology Department and encouraged by our Institution VNR Vignana Jyothi Institute of Engineering and Technology.

REFERENCES

- *https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/* [1]
- *http://www.thejavageek.com/2018/05/13/image-classification-using-cnn/* [2]
- *https://ieeexplore.ieee.org/document/8379889* [3]
- Deep Learning with Python by François Chollet [4]
- Practical Convolutional Neural Networks by Pradeep Pujari, Md. Rezaul Karim, Mohit Sewak [5]
- *https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks* [6]
- *http://ijarcsse.com/Before_August_2017/docs/papers/Volume_5/1_January2015/V5I1-0291.pdf* [7]
- *https://www.embeddedvision.com/platinummembers/cadence/embeddedvisiontraining/documents/pages/neuralnetworksimagerecognition* [8]
- *https://mafiadoc.com/using-convolutional-neural-networks-for-image-recognition_5c1a0a04097c4746268b4754.html* [9]
- *https://ip.cadence.com/uploads/901/cnn_wp-pdf* [10]
- *https://ijsr.net/archive/v4i8/SUB157179.pdf* [11]
- *http://eict.iitr.ac.in/downloads/ImagingatIITR.pdf* [12]
- *https://link.springer.com/chapter/10.1007/978-3-319-16841-8_52* [13]
- *https://www.bartleby.com/essay/Important-Technologies-Of-Dsp-And-Reason-Behind-FKYJ6MW3PV85* [14]
- *https://www.bartleby.com/essay/Important-Technologies-Of-Dsp-And-Reason-Behind-P32W6BQ3FTD5* [15]