Global Scientific JOURNALS

GSJ: Volume 8, Issue 4, April 2020, Online: ISSN 2320-9186
www.globalscientificjournal.com

Deep Learning based object detection from image using Convolutional Neural Network

Author: Iftikharul Islam*, Sujit Kumar Mondal*, Bappa Sarkar*, Joyassree Sen*

*→ Dept. of Computer Science and Engineering, Islamic University, Kushtia, Bangladesh.
For editorial correspondence:**bappacse07@gmail.com**

ABSTRACT

Object detection is a subfield of computer vision that is currently heavily based on machine learning. For the past decade, the field of machine learning has been dominated by so-called deep neural networks, which take advantage of improvements in computing power and data availability. A subtype of a neural network called a convolutional neural network (CNN) is well-suited for image-related tasks. The network is trained to look for different features, such as edges, corners and color differences, across the image and to combine these into more complex shapes.

For project, we reviewed the current literature on convolutional object detection and tested the implement ability of one of the methods. We found that convolutional object detection is still evolving as a technology, despite outranking other object detection methods. By virtue of free availability of datasets and pre-trained networks, it is possible to create a functional implementation of a deep neural network without access to specialist hardware. Pre-trained networks can also be used as a starting point for training new networks, decreasing costly training time.

Keyword: CNN, Machine learning, ANN, Convnet

1. Introduction

There is a huge amount of image data in the world, and the rate of growth itself is increasing. Before around 2012, a dataset was considered relatively large if it contained 100+ images or videos. Now, datasets exist with numbers ranging in the millions. Many of these images are stored in cloud services or published on the Internet. Over 1.8 billion images were uploaded daily to the most popular platforms, such as Instagram and Facebook.

We need to have some effective ideas about its contents to manage all of this data. Automated processing of image contents is useful for a wide variety of image-related tasks. For computer systems, this means crossing the so-called semantic gap between the pixel level information stored in the image and the human understanding of the same images. Computer vision attempts to bridge this cap. Object detection from repository of images is challenging task in the area of computer vision

1.1 Problem Statement

Image files that contain different objects can be automatically identified. This is one of the basic problems of computer vision and this is called *Object detection.* Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. As we will demonstrate, convolution neural networks are currently the state-of-the-art solution for object detection. The main task of this project is to develop and test object detection system for images based on convolution neural network.

1.2 Applications

Object detection methods frequently use extracted features and learning algorithms to recognize instances of an object or images belonging to an object category. Some applications of Object detection are listed below:

- Face detection
- Vehicle Detection
- Tracking objects
- People counting
- Scan recognition

2.1 MACHINE LEARNING

Different learning algorithms are used in computer vision applications. So before working with image related works, we are going to have a brief look at basics of machine learning. Machine learning [1] is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

In the last few years, deep convolution neural network learning has proved the outstanding performance in the field of image classification, machine learning and pattern recognition. Above all existing model, CNN is one of the most popular models and has been providing the state-of-the-art detection accuracy on object detection, segmentation, human activity analysis, image super resolution, object recognition, scene understanding, tracking and image captioning. For task of image classification CNN is outperforms above all the previous classification method. CNN extract feature from the image by a series of operations.

2.1.1 TYPES

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.

2.1.2 FEATURES EXTRACTION

In general, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Often, it is impractical or impossible to use the full-dimensional training data directly. Rather, detectors are programmed to extract interesting features from the data, and these features are used as input to the machine learning algorithm [5].

In the past, the feature detectors were often hand-crafted. The problem with this approach is that we do not always know in advance, which features are interesting. The trend in machine learning has been towards learning the feature detectors as well, which enables using the complete data.

2.1.3 GENERALIZATION

A machine learning algorithm[5] is used to fit a model to data. Once training is over, the model is unleashed upon new data and then uses what it has learned to explain that data. Now here's where problems can emerge. If we overstrain the model on the training data, then it will be able to identify all the relevant information in the training data, but will fail miserably when presented with the new data. We then say that the model is incapable of *generalizing*, or that it is *over fitting* the training data. The performance of the algorithm can be evaluated from the quality and quantity of errors. A loss function, such as mean squared error, is used to assign a cost to the errors. The objective in the training phase is to minimize this loss.

2.2 NEURAL NETWORKS

Neural networks are a popular type of machine learning model. A special case of a neural network called the convolutional neural network (CNN) is the primary focus of this thesis. Before discussing CNNs, we will discuss how regular neural networks work [4].

2.2.1 ORIGINS

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

Even though the inspiration from biology is apparent, it would be misleading to overemphasize the connection between artificial neurons and biological neurons or neuroscience. The human brain contains approximately 100 billion neurons operating in parallel. Artificial neurons[4] are mathematical functions implemented on more-or-less serial computers. Research into neural networks is mostly guided by developments in engineering and mathematics rather than biology.

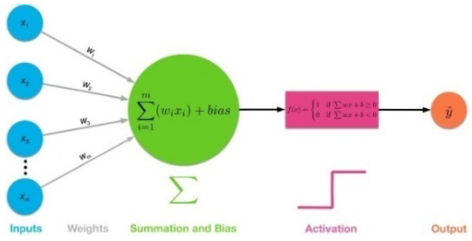

Figure 2.1: An artificial neuron

An artificial neuron based on the McCulloch-Pitts model is shown in Figure 2.1. The neuron receives m input parameters x_i. The neuron also has m weight parameters w_i. The weight parameters often include a bias term that has a matching dummy input with a fixed value of 1.
The inputs and weights are linearly combined and summed. The sum is then fed to an activation function $f()$ that produces the output ŷ of the neuron:

$$\hat{y} = f\left(\sum_{i=1}^{m}(w_i\, x_i) + bias\right)$$

The neuron is trained by carefully selecting the weights to produce a desired output for each input.

2.2.2 MULTI-LAYER NETWORKS

A multi-layer neural network is composed of one input layer, one or more layers of hidden layers and one final layer called output layer. Every layer except output layer includes a bias neuron and is fully connected to the next layer. The input layer usually merely passes data along without modifying it. Most of the computation happens in the hidden layers.

The output layer converts the hidden layer activations to an output, such as a classification When an ANN has two or more hidden layers, it is called a deep neural network (DNN). In this project, we will mostly discuss fully connected networks and convolutional networks [6].

2.2.3 BACK-PROPAGATION

The back-propagation algorithm provides a simple and effective solution to solve the weights iteratively. The classical version uses gradient descent as optimization method. Gradient descent can be quite time-consuming and is not guaranteed to find the global minimum of error, but with proper configuration works well enough in practice.

In the first phase of the algorithm, an input vector is propagated forward through the neural network. Before this, the weights of the network neurons have been initialized to some values, for example small random values.

The received output of the network is compared to the desired output (which should be known for the training examples) using a loss function. The gradient of the loss function is then computed. This gradient is also called the error value. When using mean squared error as the loss function, the output layer error value is simply the difference between the current and desired output.

The error values are then propagated back through the network to calculate the error values of the hidden layer neurons. The hidden neuron loss function gradients can be solved using the chain rule of derivatives. Finally, the neuron weights are updated by calculating the gradient of the weights and subtracting a proportion of the gradient from the weights. This ratio is called the *learning rate*; the learning rate can be fixed or dynamic. After the weights have been updated, the algorithm continues by executing the phases again with different input until the weights converge.

2.2.4 ACTIVATION FUNCTION

Activation functions are really important for an Artificial Neural Network [3] to learn and make sense of something really complicated and non-linear complex functional mappings between the inputs and response variable. They introduce non-linear properties to our network. Their main purpose is to convert an input signal of a node in an ANN to an output signal. That output signal now is used as an input in the next layer in the stack.

In artificial neural network we apply mostly RELU (not compulsory we can use different activation function) activation function which replaces the all negative values to 0 and remains same with the positive values.

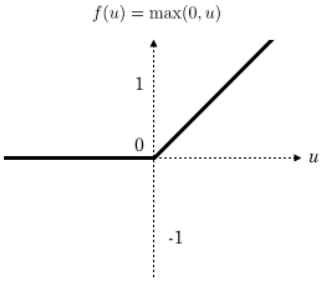

Figure 2.2: ReLu activation function

In the past, nonlinear functions like tanh and sigmoid were used, but researchers found out that relu layers work far better because the network is able to train a lot faster (because of the computational efficiency) without making a significant difference to the accuracy.

For multi-class classification problems, the softmax activation function is used in the output layer of the network:

$$f(x) = \frac{e^x}{\sum_{k=1}^{K} e^x}$$

The softmax function takes a vector of K arbitrarily large values and outputs a vector of K values that range between 0..1 and sum to 1. The values output by the softmax unit can be utilized as class probabilities [4].

2.3 COMPUTER VISION

Computer vision deals with the extraction of meaningful information from the contents of digital images or video. This is distinct from mere image processing, which involves manipulating visual information on the pixel level. Applications of computer vision include image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision and traffic automation [5].

2.3.1 OBJECT DETECTION

Object detection is one of the classical problems of computer vision and is often described as a difficult task. In many respects, it is similar to other computer vision tasks, because it involves creating a solution that is invariant to deformation and changes in lighting and viewpoint. It involves both locating and classifying regions of an image.[6].

Low-level visual features of an image, such as a saliency map, may be used as a guide for locating candidate objects. The location and size is typically defined using a bounding box, which is stored in the form of corner coordinates. Using a rectangle is simpler than using an arbitrarily shaped polygon, and many operations, such as convolution, are performed on rectangles in any case. The sub-image contained in the bounding box is then classified by an algorithm that has been trained using machine learning. The boundaries of the object can be further refined iteratively, after making an initial guess [5] [7].

2.4 CONVOLUTION NEURAL NETWORKS

Convolutional neural network (CNN or convnet) are very similar to ordinary Neural Networks. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. A simple convnet for CIFAR-10 classification could have the architecture **[INPUT - CONV - RELU - POOL - FC].**
There are four main operations in the convnet. Figure shown the basic CNN architecture for classification where the first portion describe as the feature extraction part and the next portion describe as the classification part.

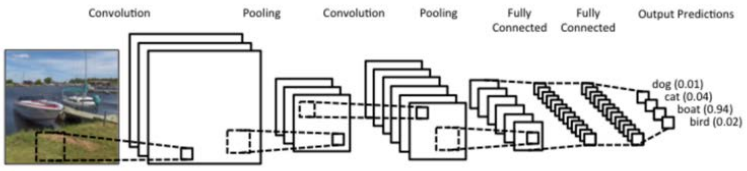

Figure 2.3**:**Basic CNN Architecture

2.5 BASIC CNN ARCHITECTURE

Convolution neural network is a sequence of layers, and every layer of a convnet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build convnet architectures:

- Convolution Layer
- Pooling Layer
- Fully connected layer

2.5.1 CONVOLUTION LAYER

The Convolution layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. It performs the convolution operations over the input volumes.
Given a two-dimensional image, **I** , and a small matrix, **K** of size **h x w** which we assume encodes a way of extracting an interesting image feature, we compute the convolved image, **I*K,** by overlaying the kernel on top of the image in all possible ways, and recording the sum of elementwise products between the image and the filter[3].

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} * I_{x+i-1,y+j-1}$$

2.5.2 Pooling and stride

Stride can be defined as the number of pixels by which slide is functional for filter matrix over the input matrix. When the stride is 1 the filters move one pixel at a time. And for the value of the stride is 2, the filters jump 2 pixels at a time. Having a larger value of stride may cause to problem smaller feature maps. As the convolutional layer does, the pooling layer also has small window (kernel). Applying the small window across images, the pooling layer conducts statistical process. The computation of the output shape after the pooling layer can be represented as the same as the convolutional layer. There are two well-used pooling layers; average pooling layer and max pooling layer. Max-pooling simply outputs the maximum value within a rectangular neighborhood of the activation map [3][8].

2.5.3 FULLY CONNECTED LAYER

The fully-connected layer contains neurons which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. This is analogous to way that neurons are arranged in traditional forms of ANN. The fully connected (FC) layer in the CNN represents the feature vector for the input. This feature vector/tensor/layer[8] holds information that is vital to the input.

Steps in the training process of CNN

Step 1: Initialize all filters and parameters/weights with random values.
Step 2: The network takes a training images as input, goes through the forward propagation step (convolution, Relu and pooling operations along with forward propagation in the fully connected layer) and finds the output probabilities for each class.
Step 3: Calculate the total error at the output layer

Total error = $\sum$ ½ (target probability –output probability)^2

Step 4: Use Back propagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.
Step 5: Repeat steps 2-4 with all images in the training set.

2.6 DATA AUGMENTATION AND DROPOUT

An over fitting model (neural network or any other type of model) can perform better if learning algorithm processes more training data. While an existing dataset might be limited, for some machine learning problems there are relatively easy ways of creating synthetic data. For images some common techniques include translating the picture a few pixels, rotation, scaling [3].

At each training iteration a dropout layer randomly removes some nodes in the network along with all of their incoming and outgoing connections. Dropout can be applied to hidden or input layer.

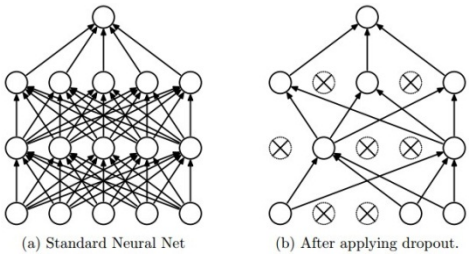

Figure 2.4: Dropout Example

The nodes become more insensitive to the weights of the other nodes (co-adaptive), and therefore the model is more robust. If a hidden unit[9] has to be working well with different combinations of other hidden units, it's more likely to do something individually useful.

3.1 Development tools and libraries

We used python programming language along with necessary development tools and different useful machine learning/deep learning libraries [10]. Python is a great general-purpose programming language on its own, but with the help of a few popular libraries it becomes a powerful environment for scientific computing. We choose python to build our model because python has many highly developed deep learning libraries which helps us building this object detection model easily and more accurate. List of used python deep learning libraries and development tools are given below:

Python libraries for deep learning:

1) Numpy
2) Tensorflow
3) Keras
4) Matplotlib

Development tools:

1) Google Colab
2) Jupyter notebook

3.1.1 Numpy

Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level basic and advancedmathematical functions to operate on these arrays. Besides its obvious scientific uses, numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows numpy to seamlessly and speedily integrate with a wide variety of databases [11].

3.1.2 Tensorflow

Tensorflow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (cpus, gpus, tpus), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains [12].

3.1.3 Keras

Keras is a high-level library that's built on top of Theano or Tensorflow. It provides a scikit-learn type API (written in Python) for building Neural Networks. Developers can use Keras to quickly build neural networks without worrying about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods. The key idea behind the development of Keras is to facilitate experimentations by fast prototyping. The ability to go

from an idea to result with the least possible delay is a key to good research. This offers a huge advantage for scientists and beginner developers alike because they can dive right into Deep Learning without getting their hands dirty with low-level computations. The rise in the demand for Deep Learning has resulted in the rise in demand of people skilled in Deep Learning [13].

3.1.4 Matplotlib

Matplotlib is a Python package for data visualization. It allows easy creation of various plots, including line, scattered, bar, box, and radial plots, with high flexibility for refined styling and customized annotation. The versatile artist module allows developers to define basically any kind of visualization. For regular usage, Matplotlib offers a simplistic object-oriented interface, the pyplot module, for easy plotting. Besides generating static graphics, Matplotlib also supports an interactive interface which not only aids in creating a wide variety of plots but is also very useful in creating web-based applications [14].

3.2 Development tools

3.2.1Google colabrotory/colab

For training model with large scale dataset requires good hardware configuration, so we have to use Google Colab. Google Colab is a free cloud service. Google Colab is a Google product for machine learning/ deep learning researchers and developers. It provides free GPU (Graphics processor unit) that very useful when we have to work for large scale dataset. It reduces the computational cost [15].

3.2.2 Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. It is a great tool for exploratory data analysis and widely used for data scientists [16].

4.1 Model design and Implementation

In this method, at first we trained our model using convolution neural network with CIFAR 10 Dataset. CIFAR 10 datasets contains 10 unique data-classes. So this model is train only for this 10 objects. Our dataset image dimension is 32x32. We use different convolution layer to extract data from input and train to fit the model.

4.1.1 Dataset description

In this paper, we used CIFAR-10 dataset to train the model. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Here are the classes in the dataset, as well as 10 random images from each:

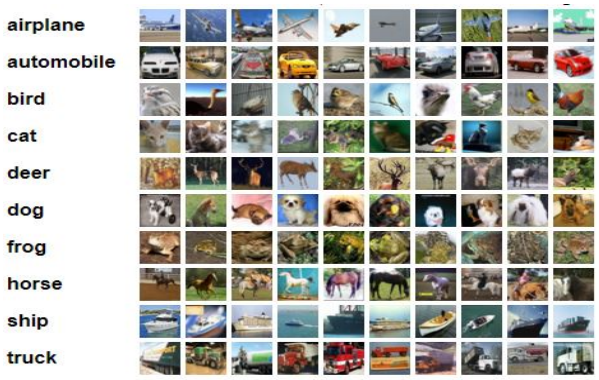

Figure 4.1: CIFAR-10 Dataset

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class [17].

4.2 The network

We trained a deep CNN with four convolutional layers and two fully connected layers. The first convolutional layer had 32 32 32 lters, the second one had 64 16 16 lters, and the last one had 128 4 4 lters.

Layer(type)	Output Shape
conv2d_1	(32, 32, 32)
max_pooling2d_1	(32,16,16)
dropout_1	(32, 16, 16)
conv2d_2	(32, 16, 16)
max_pooling2d_2	(32, 16, 16)
dropout_2	(64, 8, 8)
conv2d_3	(128, 8, 8)
max_pooling2d_3	(128, 4, 4)
dropout_3	(128, 4, 4)
flatten_1	(2048)
dense_1	(80)
dropout_4	(80)
dense_2	(10)

Table 4.1 Different layer and their dimensions used on this project

In all the convolutional layers, we have a stride of size 1, batch normalization, dropout, max-pooling and relu as the activation function. The hidden layer in the first FC layers had 80 neurons. In FC layer, same as in the convolutional layers, we used batch normalization, dropout and relu. Also we used Softmax as our loss function. Table 4.1 shows the architecture of this deep network.

4.3 Results

To show the performance of the deep CNN model, we plotted the loss history and the obtained accuracy for the model. Figures 4.2 and 4.3 exhibit the results. As seen in Figure 4.2, the deep network validation accuracy is 78.71%.

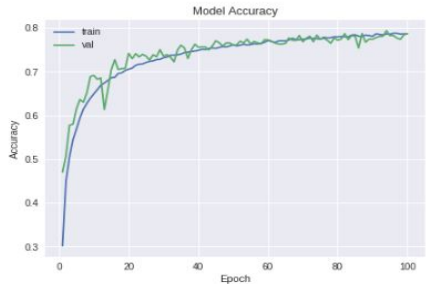

Figure 4.2: Accuracy of training and validation data set

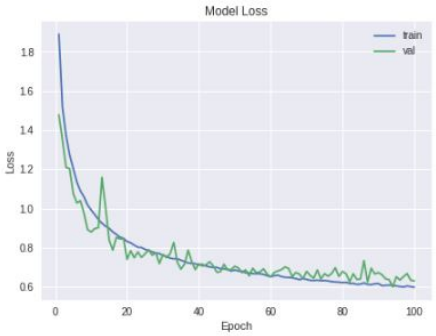

Figure 4.3: Loss history of training and validation data set

4.4 Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

We computed the confusion matrices for our deep CNN. Table 4.2 presents the visualization of the confusion matrices. As demonstrated, the deep network results in higher true predictions for most of the labels.

Null	Airplane	automobile	bird	cat	deer	dog	frog	house	Ship	truck
Airplane	599	5	74	98	55	14	12	9	117	17
automobile	16	738	12	65	9	26	7	6	40	81
Bird	31	0	523	168	136	86	33	14	9	0
Cat	10	1	31	652	90	175	19	15	5	2
Deer	6	0	34	132	717	55	16	31	9	0
Dog	5	1	17	233	53	661	10	15	4	1
Frog	2	1	39	157	105	48	637	3	7	1
House	6	0	14	97	103	96	5	637	5	1
Ship	41	7	28	84	19	18	6	4	783	10
truck	25	28	8	77	29	27	5	19	59	723

Table 4.2: Confusion Matrix for object detection system

In the future work, we would like to plan to use more advanced network that will be helpful to train deep architectures and allow us to investigate the accuracy of our object detection system. Image localization was left outside the scope of the paper due to not having GPU.

4.6 Bibliography

[1]https://expertsystem.com/machine-learning-definition/
[2] Bishop, C. M. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[3]Goodfellow, I., Bengio, Y., and Courville, A. Deep Learning. MIT Press, 2016
[4]Long, L. N., and Gupta, A. Scalable massively parallel artificial neural networks. Journal of Aerospace Computing, Information, and Communication 5, 1 (2008)
[5]Szeliski, R. Computer vision: algorithms and applications. SpringerScience & Business Media, 2010.
[6]Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (2015), pp. 1440-1448.
[7] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation.
[8]Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (2015), pp. 91-99.
[9]Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15 (2014), 1929-1958.
[10]https://www.python.org/
[11]https://numpy.org/
[12]https://www.tensorflow.org/
[13]https://keras.io/
[14]https://matplotlib.org/
[15]https://colab.research.google.com/
[16] https://jupyter.org/
[17]https://www.cs.toronto.edu/~kriz/cifar.html