Eng. Sci., 2021, **15**, 67–79

Engineered Science

DOI: https://dx.doi.org/10.30919/es8d461

The Effect of High-Performance Computer on Deep Neural Network

Xin Zhang,[1] Ting Zhang,[2] Jiang Lu,[1*] Xingang Fu[3] and Francisco Reveriano[1]

Abstract

In recent years, convolutional neural networks (CNNs) have been extended widely to a large number of computer vision applications such as image classification, image detection, image segmentation, *etc*. In this paper, the three image processing applications are implemented by integrating with CNNs and the high performance computing (HPC) systems. To observe the performance of HPC, three CNN models for each image processing application have been trained with different values of parameters and their training times are provided in results. Four computing systems, Google collaboratory with central processing unit (CPU), Google collaboratory with graphics processing unit (GPU), Google Cloud with HPC, and The extreme science and engineering discovery environment (XSEDE) with HPC are compared in the work. The training program is suggested to use the parallel algorithm when GPU is available. This project explores that the HPC with GPU has the highest work efficiency regarding operating time.

Keywords: High Performance Computing, CNN, GPU.
Received: 1 March 2021; Accepted date: 3 May 2021.
Article type: Research article.

1. Introduction

Nowadays, convolutional neural networks (CNNs) have become the most important tool in image processing applications. Many very deep CNNs have already been designed. However, to implement the CNNs for image processing applications, computing resource is an extreme challenge. In the past 20 years, the computational capability and speed of computers are two of the most important parameters in computer development. So far, many advanced technologies allow computers to run faster with stronger computing capabilities such as central processing unit (CPU) with multiple cores, graphics processing unit (GPU), and computer clusters. They could change the game in such fields as computer vision, artificial intelligence (AI), cryptography, chemistry, biology, and physics. There are several popular computation resources for computers nowadays as follow:

CPU is called a central processing unit and it is often used to perform arithmetic and logic computations and acts as the brain of the computer. Modern CPUs offer multiple cores. Nowadays, CPUs can have around 2 to 18 cores. The CPU with multiple cores speeds up the system because the computer can do multiple things at once. There are some excellent multi-core processors from 2017 to 2019. For example, the Intel Core i9-7900X processor and AMD Ryzen Threadripper processor are the two most advanced multi-core processors in their product lines, respectively.[1] The computers can process complex calculations depend on Multi-cores CPU. Besides, Multi-cores CPU supports many consumer software with multithreading. GPU is known as a graphical processing unit and it is used to operate digital images as a graphical device. GPUs start at a couple of hundred cores and can have up to several thousand. The memory size of GPUs is suitable for huge amounts of data computations in deep learning. GPUs have perfect achievement in practical applications. For example, eBay's maxDNN has already shown that all of the programs can achieve excellent performance with very high GPU utilization.[2] Google Colab is a free cloud service that can provide convenient CPU and GPU resources for programmers. Google Colab is essentially a Jupyter Notebook within the

[1] *Department of Computer Engineering, University of Houston-Clear Lake, Houston, USA.*
[2] *Computer Science & Engineering Technology, University of Houston-Downtown, Houston USA.*
[3] *Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, USA.*
**Email:* luj@uhcl.edu (J. Lu)

web. The programmers can write programs and run them using a remote CPU or GPU in Google Colab. Google Colab has installed many deep learning libraries such as PyTorch, Keras, TensorFlow, and OpenCV. It's very convenient for image processing applications.

High performance computing (HPC) is an indispensable tool. An HPC system is described by numerous processors, heaps of memory, fast systems administration, and expansive information stores. The HPC is intended to utilize parallel computing to apply more processor force for the solution of a problem. Therefore, An HPC cluster is comprised of numerous nodes. The nodes include computing nodes and master nodes. Most of the nodes are compute nodes. A compute node performs one or more tasks based on the scheduling system. The master nodes observe the status of individual nodes and issue administrative orders. The extreme science and engineering discovery environment (XSEDE) can provide high-performance computing resources for scholars and researchers. XSEDE is the most advanced and powerful collection of data resources and services in the world. The data resources include supercomputers, software, networks, and data storage. XSEDE has many partner institutions including the pittsburgh supercomputing center (PSC) at the Carnegie Mellon University and the University of Pittsburgh, texas advanced computing center (TACC) at the University of Texas, Austin, san diego supercomputer center (SDSC) at the University of California and so on.[3] Google cloud is another good provider for HPC. It is applied in large data, artificial intelligence, and other fields, gradually shifting from scientific research to commercialization.[4] For example, PayPal serves more than 300 million customers and developing online, mobile, and in-store services by HPC on google cloud.

In this paper, the three basic image processing applications, image classification, image detection, image segmentation, based on different CNN architectures are tested utilizing several different computing systems. The first image processing application is image classification. Three popular deep neural networks, VGG16, ResNet50, and Inception v3 are chosen. The second image processing application is image object tracking. The three classical CNN architectures, Single Shot MultiBox Detector (SSD), Fast R-CNN, and Yolov3 are chosen. The third image processing application is image segmentation. The three popular network models are picked. They are U-net, Mask R-CNN, and PANet.

The rest of the paper is organized as follows. Section 2 briefly reviews related background. Section 3 details deep neural network structure methods and algorithms of the three image processing applications. The results are provided in Section 4. Finally, Section 5 concludes the paper.

2. Related Work

Image classification is the most common application of a convolutional neural network. There are many popular CNNs for image classification and detection. A. Krizhevsky *et al*.[5] proposed a large, deep convolutional neural network called AlexNet for image classification and detection. K. Simonyan *et al*.[6] proposed the Very Deep Convolutional Networks called VGG for the large-scale image recognition setting. C. Szegedy *et al*.[7] proposed a deep convolutional neural network architecture is called GoogleNet, which was responsible for setting the new state-of-the-art for classification and detection. K. He *et al*.[8] proposed a residual learning framework for image recognition.[9] F. Chollet[10] proposed a novel deep convolutional neural network architecture is called Xception for a larger image classification dataset.

Image detection is another important application of a convolutional neural network. So far, more and more CNN architectures support systems to track objects from an image or video. J. Redmon *et al*.[11] proposed a new approach is called YOLO to spatially separated bounding boxes and associated class probabilities. W. Liu *et al*.[12] proposed a Single Shot MultiBox Detector method for detecting objects in images using a single deep neural network. R. Girshick *et al*.[13] proposed an R-CNN method where we use selective search to extract just 2000 regions from the image. S. Ren *et al*.[14] proposed a state-of-the-art object detection network is called Faster R-CNN depend on region proposal algorithms to hypothesize object locations. J. Redmon *et al*.[15] proposed an update method of YOLO is called YOLOv3 for image object tracking.

Image segmentation is applied to machine vision, medical imaging, and video surveillance, and so on. Until now, CNN is still one of the best technologies for image segmentation. J. Long *et al*.[16] proposed a Fully Convolutional Network (FCN) (containing only convolutional layers) trained end-to-end for image segmentation. G. Sharma *et al*.[17] proposed an end-to-end convolutional network which is called ParseNet predicting values for all the pixels at the same time for image segmentation. O. Ronneberger *et al*.[18] proposed a neural network called U-net composed in the contracting part and expanding part for biological microscopy image segmentation. T.-Y. Lin *et al*.[19] proposed a feature pyramid network (FPN) for object detection or image segmentation. H. Zhao *et al*.[20] proposed the Pyramid Scene Parsing Network (PSPNet) to better learn the global context representation of a scene. K. He *et al*.[21] proposed the Mask R-CNN model for object instance segmentation. L.-C. Chen *et al*.[22] proposed the Deeplabv3+ framework using an encoder-decoder structure for image segmentation. S. Liu *et al*.[23] proposed the Path Aggregation Network (PANet) based on the Mask R-CNN and the FPN frameworks for image segmentation. H. Zhang *et al*.[24] proposed a Context Encoding Network (EncNet) capturing global information in an image to improve scene segmentation.

3. CNN Architectures for Image Processing Applications

3.1. CNN network for image classification

Image classification is the most basic application of the convolutional neural network (CNN) technology.[25] The CNN allows the computer to operate in a self-learning mode to classify multiple objects of an image, without being explicitly

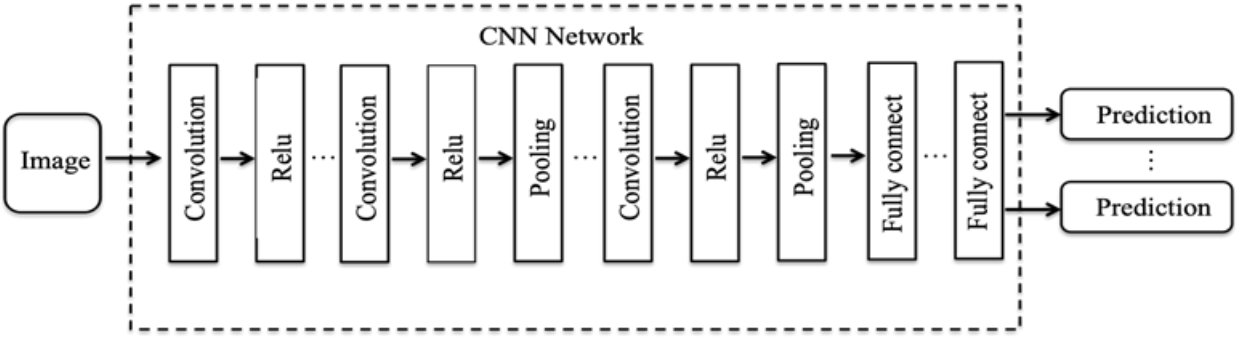


Fig. 1 A basic CNN architecture.

programmed. A CNN takes a picture that includes multiple objects, extracts features from the picture through different kinds of layers, and predicts the probability of the classes. A basic CNN architecture has an input layer, convolutional layers, Relu layers, pooling layers, and a fully connected layer. Fig. 1 shows a basic CNN architecture.

The input layer reads an array of pixels from an input image. For example, if the size of an image is 256 × 256 × 3. Where the first 256 is width, the second 256 is height, and 3 is RGB channel values. Therefore, this image has a total of 196,608 pixels. The value of each pixel has a number from 0 to 255 which is the intensity of each pixel.

The convolutional layer is the key technology of CNN. When the matrix with pixel values entered into the convolutional layer, the network began reading pixel values from the top left of the matrix. The network selected a small filter that moves along the matrix and operates these pixel values in the filter. The filter multiplies its value by the original pixel values and then sums up all these multiplications. After passing the filter across all positions of the matrix, the network obtained a new matrix that is smaller than the input matrix.

The Relu layer is added after the convolutional layer. The Relu layer has a Relu activation function that helps to decide if the neuron would work or not. ReLU function is the most widely used activation function in CNN. It converts all negative inputs to zero and the neuron does not get activated. Therefore, the Relu function generates the variable to decide a class label.

The pooling layer follows the Relu layer. The target of the pooling layer is to reduce the number of parameters and computation in the network. Because some features have been identified in the previous convolutional layer, the pooling layer compressed these features for controlling overfitting.

The fully connected layer is the last layer in CNN. It takes the output information from convolutional networks, flattens them, and turns them into a single vector. It reaches a classification decision for the correct label.

This paper chose three kinds of common deep CNN architectures to test the performance of the effect of high-performance computer. These architectures are VGG16, Resnet50, and Inception v3. Each architecture has its self-characteristics to be applied to different fields and hardware environments. For example, VGG is now one of the most used image-recognition architectures. Resnet is one of the most popular architectures in various computer vision tasks. Inception is applied to mobile and embedded system environments.

3.1.1 VGG network architecture

VGG network is a very deep CNN for large-scale image recognition. VGG network used only a 3 × 3 filter in the first layer network. The advantage of a small filter is to reduce the number of parameters and allows VGG to have a large number of weight layers. Reducing the number of parameters decreases the complexity of the network. So When the VGG network has very deep layers, it still can handle overfitting. The convolutional layers in VGG are followed by a Relu unit. VGG has three fully-connected layers: the first two have 4096 channels each and the third has 1000 channels, 1 for each class. Based on the depth of the network, the family of VGG networks includes VGG11, VGG13, VGG16, VGG19. Fig. 2 shows a basic VGG network architecture.[26]

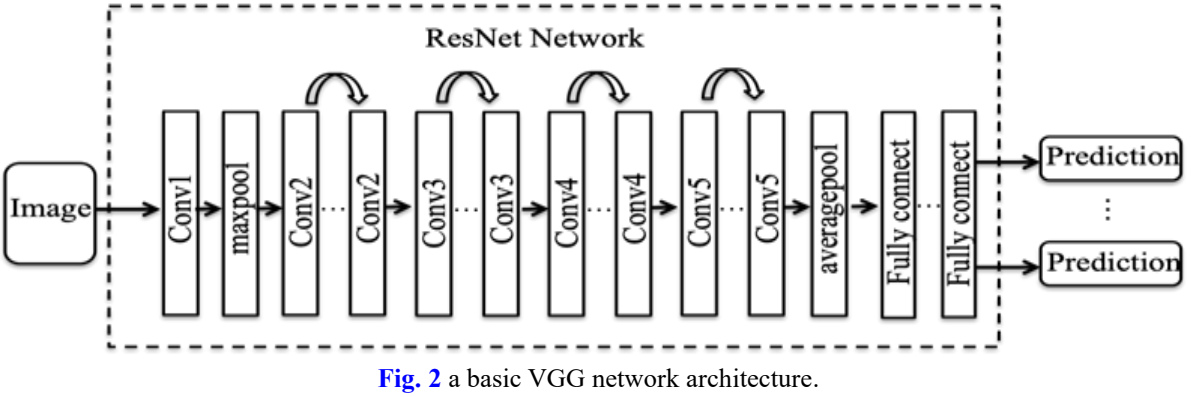


Fig. 2 a basic VGG network architecture.

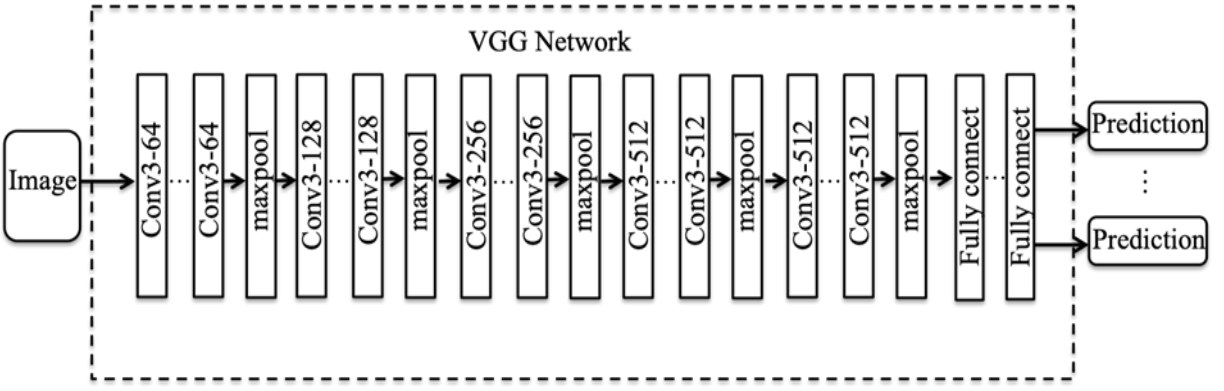


Fig. 3 a basic ResNet network architecture.

3.1.2 ResNet network architecture

ResNet network can build a very deep CNN that is over a hundred layers by learning the residual representation functions. ResNet network solves the vanishing gradient problems when the CNN architecture is going deeper and deeper. The key technology of ResNet network is called "identity shortcut connection" which skips one or more layers. The shortcut connection is added from the input value to the output value after few convolutional layers. Resnet network uses zero-padding and a linear projection can handle the problem that input and output have a different size at the shortcut connection. ResNet network includes many residual blocks. Each residual block has a shortcut connection and several convolutional layers. ResNet network consists of ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152. Fig. 3 shows a basic ResNet network architecture.[27]

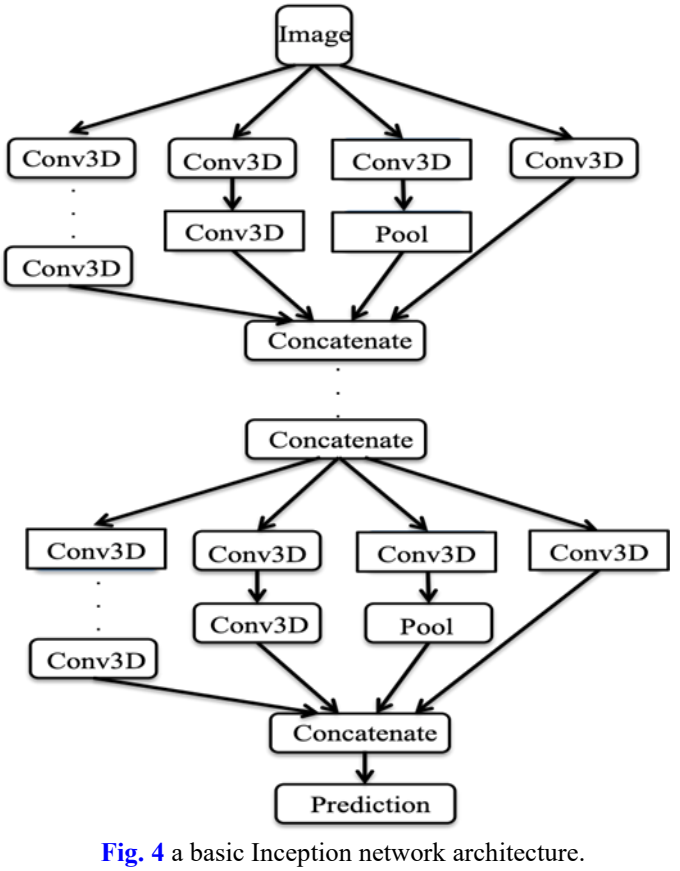


Fig. 4 a basic Inception network architecture.

3.1.3 Inception network architecture

The inception network is a depthwise separable convolutional network by Google. The inception network is also a pretty deep network that is subject to the vanishing gradient problem. The inception network includes many inception modules. Each inception module uses a different size filter to capture different scale information on parallel convolutional layers. For example, a small-size filter focus on detail like dense contours, and large-size filter benefits for processing coarse outlines. Then all outputs from these parallel convolutional layers are concatenated together and sent to a fusion layer. The concatenated features in the fusion layer are fed to the next inception module as the input. The inception network includes inception v1, inception v2, inception v3, and inception v4. Fig.4 shows a basic Inception network architecture.[28]

3.2. CNN network for image detection

Image classification is the more advanced application of the convolutional neural network (CNN) technology than image classification. The image detection network is divided into two parts: the backbone network determines the classification of objects and the object detector network determines the location of objects. The backbone network is a deep neural network that extracts the basic features for object detection. The object detector network is a single deep neural network that predicts the bounding boxes and the class probabilities for all detecting objects. Generally, the object features are extracted from the input image using the backbone network at first. Then the output layers of the backbone network connect the object detector network for the classification of the extracted features. The output of the object detector network classifies N + 1 predictions, where N is the number of classes, and 1 is for the background. The output also provides 4 coordinates predictions of each bounding box. The common image detection network architectures include SSD, R-CNN, and YOLO. The paper chose these three kinds of common image detection architectures to test the performance of the effect of HPC. Fig.5 shows the architecture of a general object detection network.

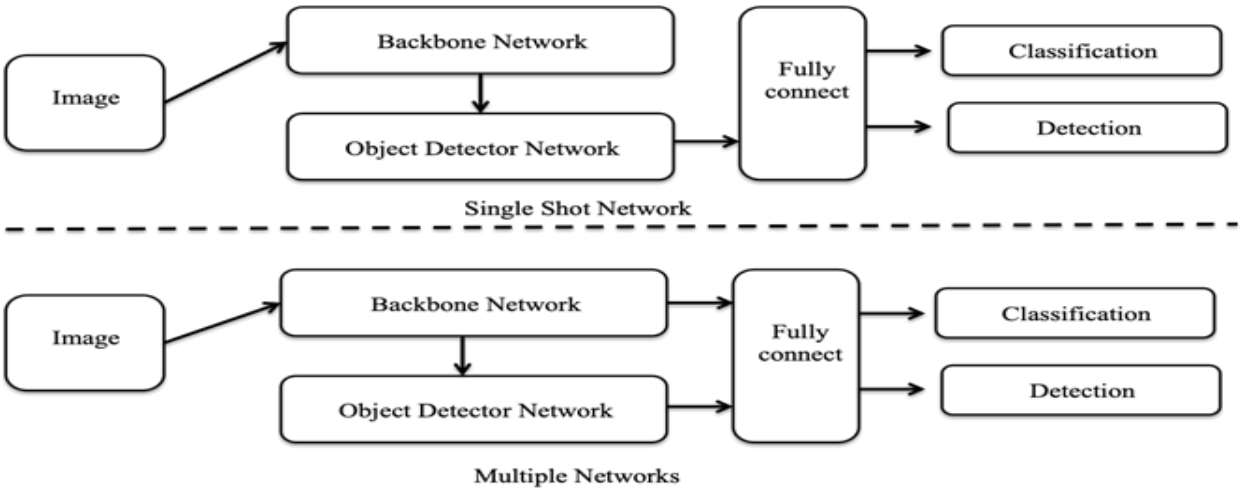


Fig. 5 The architecture of a general object detection network.

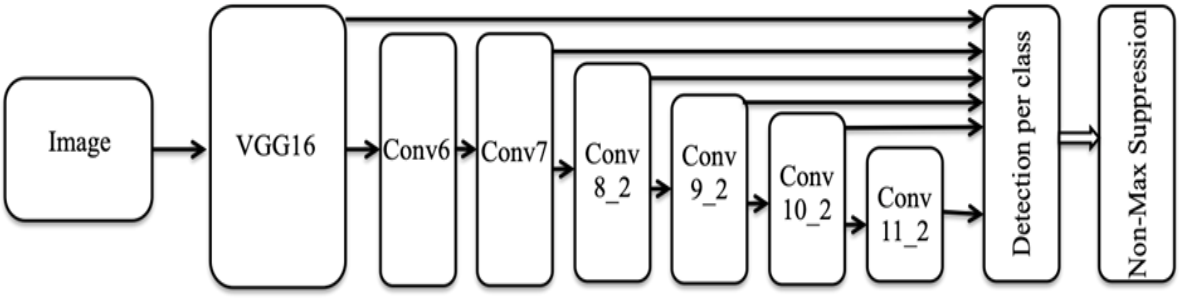


Fig. 6 SSD network architecture.

3.2.1 SSD network architecture

SSD network is a single shot mutibox detector for object detection in real-time. SSD network uses VGG16 network as the backbone network. VGG16 discards the fully connected layers and adds 6 auxiliary convolutional layers as the object detector network. Therefore, the SSD network can use multiple layers to detect objects independently. The auxiliary convolutional layers are for object detection. Multi-scale feature maps can improve accuracy significantly. SSD network associates a set of default bounding boxes with each auxiliary convolutional layer. The default bounding boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. Object detector network predicts 4 offsets relative to the original default box shape depend on feature map cells. Fig. 6 shows SSD network architecture.[29]

3.2.2 Fast R-CNN network architecture

THE fast Region Based Convolutional Neural Networks (R-CNN) network is designed to tackle object detection problems. Fast R-CNN network includes backbone network, region proposal network and full connect network. The backbone network extracts the features for object classification. The region proposal network is similar to the backbone network. It combines the features and forms a fixed-length feature vector in the RoI pooling layer. Each of the feature vectors consists of a classification module and a localization module. The classification module classifies object classes. The localization modules output four locations for each object class. The full connect network connects the RoI pooling layer for the classification and localization of objects. Fig. 7 shows Fast R-CNN network architecture.[30]

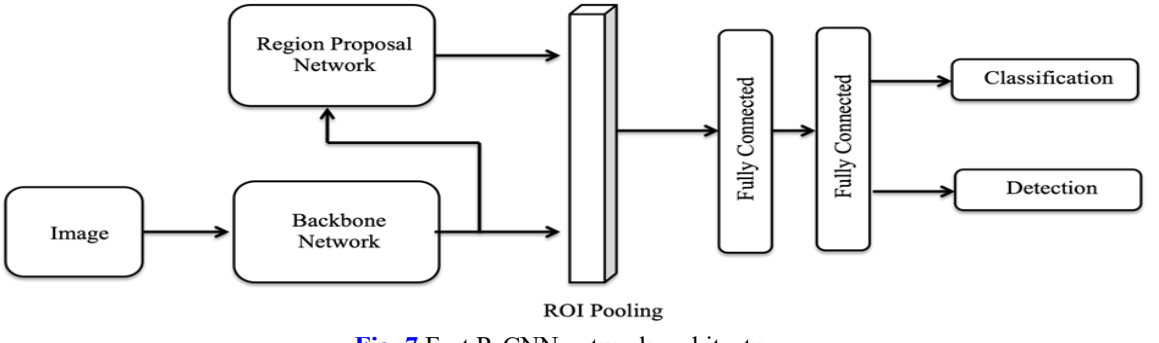


Fig. 7 Fast R-CNN network architecture.

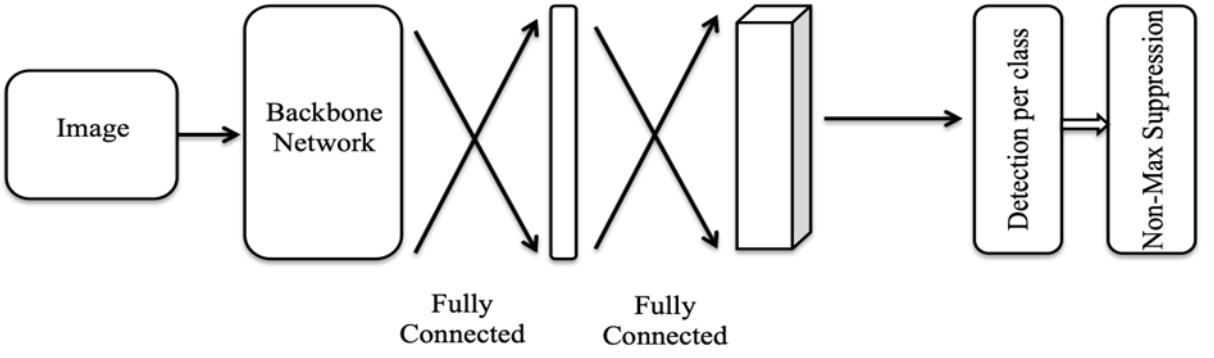


Fig. 8 YOLO network architecture.

3.2.3 YOLO network architecture

YOLO network is one of the faster CNN networks for object detection. Although it is not the most accurate object detection network, it is a good choice for real-time detection without loss of too much accuracy. Like the SSD network, the YOLO network also uses a deep CNN network as the backbone network for feature extraction and an FPN network for object detection. YOLO network algorithm splits an input image into m x m grid cells. Each grid cell predicts whether the center of the object falls into the grid cell. YOLO networks include YOLO v1, YOLO v2, YOLO v3, YOLO v4, and YOLO v5. Fig. 8 shows YOLO network architecture.[31]

3.3. CNN network for image segmentation

Image segmentation is one of the most fast-growing applications of the CNN network because artificial intelligent machines need to analyze any object in a given image scenario today. It needs to combine image classification and detection technologies. Image segmentation is of two types: semantic segmentation and instance segmentation. Semantic segmentation links each pixel for each class label in an image. U-Net networks are for semantic segmentation. Instance segmentation masks each instance of an object contained in an image independently. Mask R-CNN network is for instance segmentation.

The panoptic segmentation combines semantic and instance segmentation such that all pixels are assigned a class label and all object instances are uniquely segmented. PANet network is for panoptic segmentation. The paper chose these three kinds of common image segmentation architectures to test the performance of the effect of HPC.

3.3.1 U-net network architecture

U-net network looks like a "U" which justifies its name. U-net is one of the famous Fully Convolutional Networks (FCN) for biomedical image segmentation. FCN network is an end-to-end deep CNN network for the prediction of image segmentation. FCN network is different from traditional CNN. It gets rid of fully-connected layers and only uses convolution and pooling layers. U-net network consists of three parts: contraction network, bottleneck network, and expansion network. The contraction network is made of many convolution blocks. Each block has two convolution layers with 3 × 3 filters followed by a max-pooling layer with 2 × 2 filters. It tries to extract the features from the input image with a series of convolution layers. The bottleneck network connects the contraction network and the expansion network. It uses two convolution layers with 3 × 3 filters followed by an up convolution layer with 2 × 2 filters. The extension network is similar to the contraction network. It also has many convolution blocks. Each block has two convolution layers with 3 × 3 filters followed by a 2 × 2 upsampling layer with 2 × 2 filters. An extension network is used to reconstruct the features. Fig. 9 shows U-net network architecture.

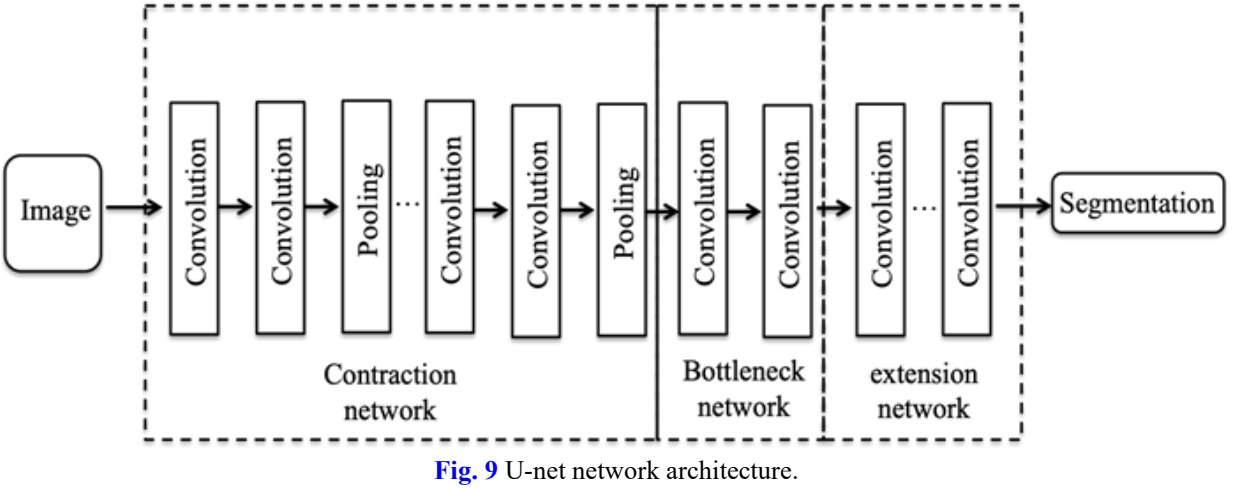


Fig. 9 U-net network architecture.

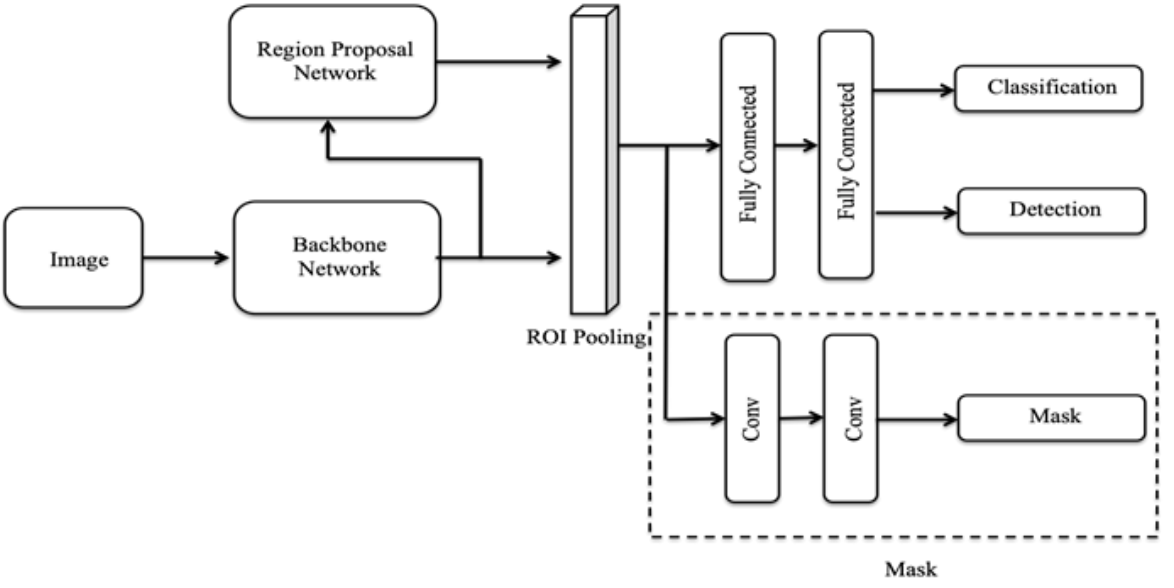


Fig. 10 Mask R-CNN network architecture.

3.3.2 Mask R-CNN network architecture

Mask R-CNN network is a deep neural network based on the Faster R-CNN network for instance segmentation. Fast R-CNN network classifies the objects and finds the bounding box of each object from an image. Extending the Faster R-CNN network, the Mask R-CNN network adds a binary mask classifier to predict a binary mask for each RoI. The binary mask classifier consists of CNN networks. This classifier uses various blocks of convolution and max pool layers to decompress an image. It then makes a class prediction at this level of granularity. Finally, it uses up-sampling and deconvolution layers to resize the image to its original dimensions. So besides object classification and object localization, the Mask R-CNN network also predicts the pixels of each object detected. Fig. 10 shows Mask R-CNN network architecture.

3.3.3 PANet network architecture

PANet network is an extended Mask R-CNN for panoptic segmentation. Usually, Mask R-CNN provides good results on instance segmentation tasks. PANet uses FPN to provide information propagation paths. FPN used employs a top-down path to combine semantically rich features from high-level layers with accurate localization information residing in the higher resolution feature-maps of lower layers. It uses adaptive feature fooling to capture information from all levels. Fig. 11 shows PANet network architecture.

4. Experiment and Results

4.1. Hardware

Google Collaboratory (Colab)[32] is a compiler tool for machine learning developers. Google Colab provides CPU and GPU for the notebook that runs the programs. In Google Colab, CPU uses 2 cores of Intel(R) Xeon(R) CPU @ 2.30GHz. The users can get 34 GB of available RAM from Google Colab. The runtime duration can stay connected for up to 24 hours, and idle timeouts are relatively lenient. GPU uses Nvidia Tesla P100. Tesla P100 is a professional graphics card by NVIDIA. It has 3584 CUDA cores and 16 GB HBM2 memory at 732 GB/s. Single-precision performance can arrive at 9.3

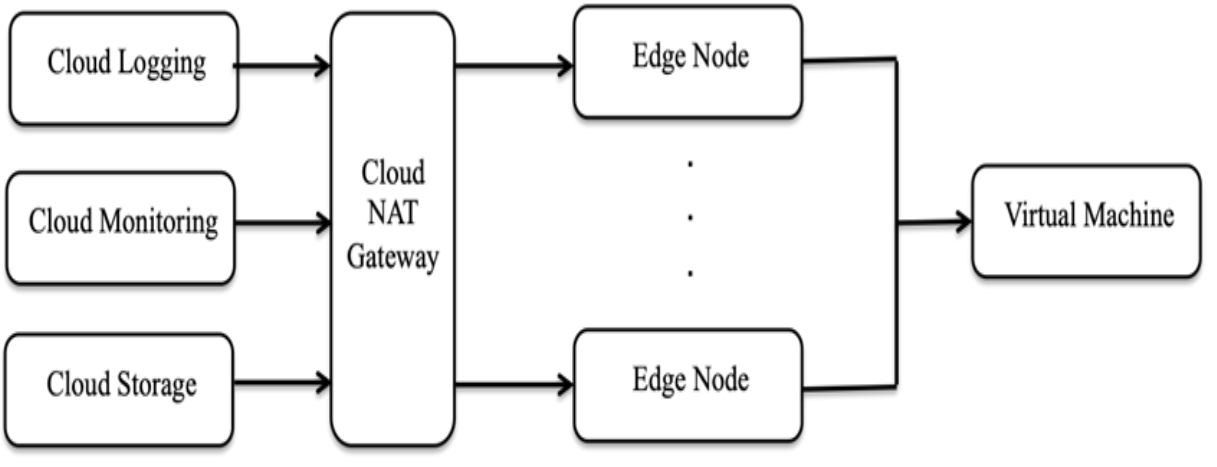


Fig. 11 PANet network architecture.

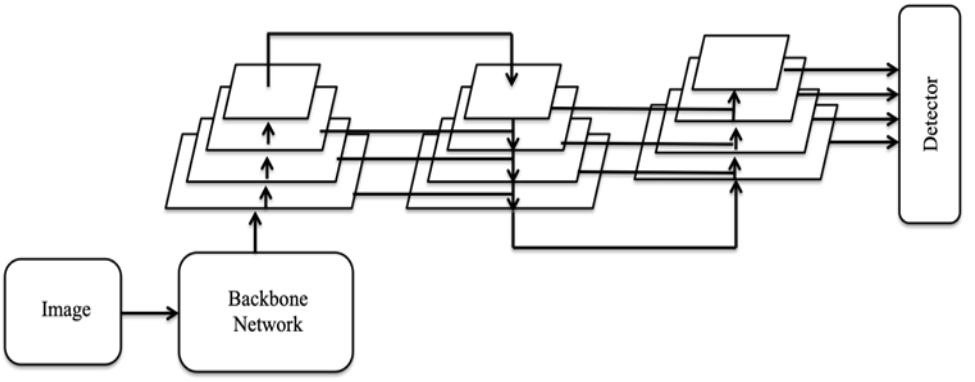


Fig. 12 HPC architecture on Google cloud.

TeraFLOPS. In our experiment, we use the CPU and GPU of Google Colab to test the performance of three image processing applications respectively.

Google cloud provides tightly coupled HPC workloads for the customers. Fig. 12 shows HPC architecture on Google cloud. Google Cloud can create a virtual machine (VM) that includes an operating system, microprocessor, memory, and storage. The VM is a custom Slurm cluster on the Google Cloud Platform. Slurm is one of the leading workload managers for HPC clusters. Slurm provides an open-source, fault-tolerant, and highly-scalable workload management and job scheduling system. In the VM, the users can customize the number of CPU cores, memory, the number of GPUs, and the GPU type one would like their virtual machine to have. In our experiment, we use the AI platform notebook provided by the Google cloud platform (GCP). The advantage of the GCP notebook is that people could edit their machine size to fit their requirements. In our experiment, we use HPC with GPU of Google Cloud to test the performance of three image processing applications respectively. In Google Cloud, GPU uses Nvidia Tesla P100. We use 4 GPUs and 64 GB HBM2 memory.

XSEDE provides HPC resources for the programmers. We can access the bridge through XSEDE. The Bridges supercomputer comprises over 850 computational nodes. The paper was running in the Regular Shared Memory GPU (RSM-GPU) nodes which contained 48 nodes running either the Tesla K80 GPUs or P100 GPUS. The server being used is called the HPE Apollo 2000. Each Bridges' GPU node has two dual GPUs and two CPUs. The program can use anywhere from one GPU on one node to all GPUs on all GPU nodes.

4.2. Image Classification

4.2.1 Software

PyTorch is an open-source machine learning library for Python. It gains widespread adoption because of its elegance, flexibility, speed, and simplicity. The PyTorch framework can be easy to build a simple neural network for an image classification problem. PyTorch provides many functions for operating on tensors. The functions keep track of all the operations performed on tensors. Therefore, tensors can accelerate the numeric computations on GPU. Pytorch contains the model architectures for image classification. The model architectures include AlexNet, VGG, ResNet, Inception v3, GoogleNet, MobileNet, and so on. Our experiment uses VGG16, ResNet18, and Inception v3 model architectures on PyTorch for image classification.

4.2.2 Dataset

In the image classification application, we use the ImageNet dataset organized according to the WordNet hierarchy. The ImageNet dataset is a large collection of human-annotated photographs for ILSVRC competition. There are more than 14 million images in the dataset and more than 21 thousand classes. We use about 1500 images for the training dataset, about 500 images for the testing dataset. In the training dataset, 1000 images are trained and 500 images are validated. We predict 10 object classes from the dataset. In the experiment, a batch size of 32 is chosen and the number of epochs is set to 100.

4.2.3 Experiment

The experiment chooses three CNN networks for image classification: VGG16, ResNet18, and Inception v3. VGG16 has 13 convolutional layers, 5 max pool layers, and 3 fully connected layers. It includes a total of 138 million parameters. All of the convolution kernels are of size 3 × 3 and max pool kernels are of size 2 × 2 with a stride of 2. ResNet18 has 17 convolutional layers, 2 pooling layers, and 1 fully connected layer. It includes a total of 11 million parameters. It consists of convolutional layers with filters of size 3 × 3. Inception v3 has 81 convolutional layers, 15 pool layers, and 4 fully connected layers. It includes a total of 24 million parameters. It consists of convolutional layers with filters of size 1 × 1, 1 × 3, 3 × 3, 1 × 7, and 7 × 7. Table 1 shows the introduction of the pre-trained model.

The experiment uses the three CNN networks to train the model on three computing platforms respectively. The input image is RGB formats and the size of the image is 224 × 224. In the first experiment, the accuracy of VGG16 is 67%, the accuracy of ResNet18 is 77%, and the accuracy of Inception v3 is 72%. The result of the experiment shows Inception v3 spends the shortest time for training the model. Although it has the most convolutional layers, the architecture of Inception

Table 1. The introduction of the pre-trained model.

Pretrained Model	Convolutional layers(levels)	Pooling layers (levels)	Fully Connected layers (levels)	Parameters(million)
VGG16	13	5	3	138
ResNet18	17	2	1	11
Inception V3	56	15	4	24

v3 is parallel. Many convolutional layers can run on the computing platform simultaneously. VGG16 spends the longest time because it has the most parameters than other networks. Compare four computing platforms, the work efficiency of high-performance computer with GPU is lower than Google Colaboratory(Colab) with CPU and GPU. Compare CPU and GPU, GPU decreases 93% than CPU in time. Fig. 13 shows the result of the training model in three architectures.

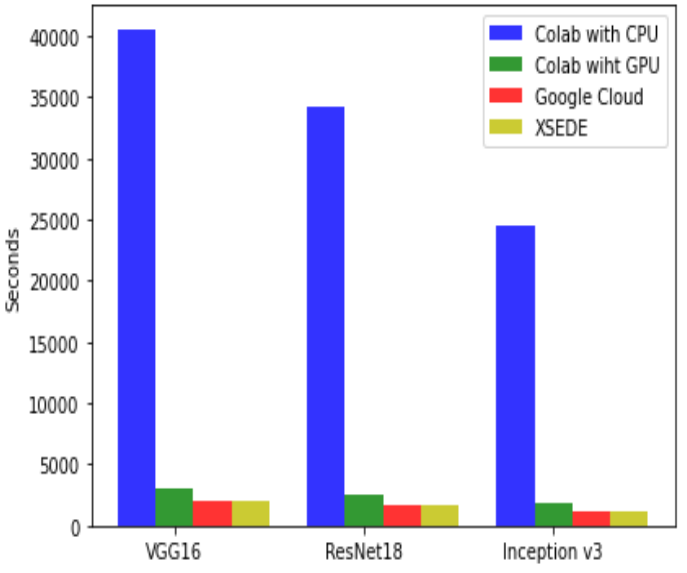


Fig. 13 The result of the training model in three architectures.

4.3. Image Detection

4.3.1 Software

TensorFlow is a computational framework for image detection. TensorFlow was developed by Google and it's one of the most popular Machine Learning libraries on GitHub. The core data type in TensorFlow is the computational graph. The nodes of the same level in a computational graph can be executed in parallel. Tensorflow allows users to make use of parallel computing devices to perform multiple node operations. It can create multiple workers to schedule tasks on various computing devices. Therefore, TensorFlow can schedule the tasks on parallel computing devices (GPU). It also can schedule the operations on CPU and GPU simultaneously. Our experiment adopts three image detection architectures with Tensorflow.

4.3.2 Dataset

In the image detection application, we use an open image dataset from the google website. It uses almost 9 million URLs for images. These images have been annotated with image-level labels bounding boxes spanning thousands of classes. The dataset contains a training dataset of 9 million images, a validation dataset of 41,260 images, and a test dataset of 125,436 images. We use about 1500 images for the training dataset, about 500 images for the testing dataset. In the training dataset, 1000 images are trained and 500 images are validated. We detect 5 object classes from the dataset. In the experiment, a batch size of 32 is chosen and the number of epochs is set to 100.

4.3.3 Experiment

The experiment chooses three CNN network architectures for image detection: SSD, Fast R-CNN, and Yolo v3. SSD network architecture has a VGG16 network as the backbone network and 6 convolutional layers as the object detector network. It includes a total of 26.3 million parameters. It consists of 19 convolutional layers. Fast R-CNN network architecture has a VGG16 network as the backbone network and a region proposal network as the object detector network. It includes a total of 134.7 million parameters. It consists of 21 convolutional layers. Yolo v3 network architecture has a darknet-53 network as the backbone network and 3 convolutional layers as the object detector network. It includes a total of 61 million parameters. It consists of 53 convolutional layers. Table 2 shows the introduction of the pre-trained model.

Table 2. The introduction of the pre-trained model.

Petrained Model	Convolutional layers(levels)	Backbone Network	Parameters(million)
SSD	19	VGG16	26.3
Fast R-CNN	21	VGG16	134.7
Yolo v3	53	Darknet-53	61

The experiment uses the three CNN architectures to train the model on three computing platforms respectively. The input image is RGB formats and the size of the image is 224 × 224. In this experiment, we see that the accuracy of SSD is 72%, the accuracy of Fast R-CNN is 83%, and the accuracy of Yolo v3 is 79%. The result of the experiment shows SSD architecture spends the shortest time for training the model because it has minimum parameters and convolutional layers. Fast R-CNN spends the longest time because it has the most parameters than other networks and two layers structure of CNN networks. Compare four computing platforms, the work efficiency of high-performance computer with GPU is lower than Google Colaboratory (Colab) with CPU and GPU. Compare GPU and Google cloud with HPC, HPC decreases

33% than GPU in time. Fig.14 shows the result of the training model in three architectures.

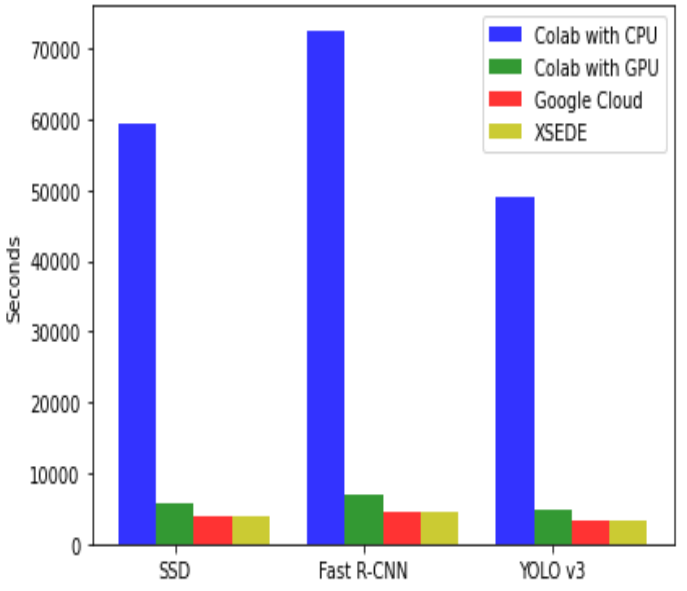


Fig. 14 The result of the training model in three architectures.

4.4. Image Segmentation

4.4.1 Software

Keras is one of the most popular deep learning libraries. It provides a convenient way to train a deep learning model. Because Keras is a neural network API that runs on top of Tensorflow, Theano, and CNTK library, the developers can be easy to create a CNN with the functional API. Keras models accept three formats of input data: NumPy arrays, TensorFlow Dataset objects, and Python generators. These input data can be preprocessed asynchronously on the CPU while your GPU is busy. The data is buffered into a queue. When GPU finished the previous batch data, the data on memory is immediately available. So GPU can reach full utilization. Our experiment adopts three image Segmentation architectures with Keras.

4.4.2 Dataset

In an image segmentation application, we use a COCO-Stuff dataset for image segmentation. COCO-Stuff dataset augments 164,000 images of the popular COCO dataset with pixel-level stuff annotations. The dataset contains a training dataset of 164,000 images, a validation dataset of 5,000 images, and a test dataset of 20,000 images for the image segmentation challenge. It covers 172 classes: 80 thing classes, 91 stuff classes, and 1 class 'unlabeled'. We use about 1500 images for the training dataset, about 500 images for the testing dataset. In all of the training datasets, 1000 images are trained and 500 images are validated. In the experiment, A batch size of 32 is chosen and the number of epochs is set to 100.

4.4.3 Experiment

The experiment chooses three CNN network architectures for image segmentation: U-net, Mask R-CNN, and PANet. U-net network architecture has 23 convolutional layers and 4 pooling layers. It includes a total of 7,759,521 parameters. All of the convolution kernels are of size 3x3 and maxpool kernels are of size 2 × 2. Mask R-CNN network architecture has a VGG16 network as the backbone network and a region proposal network as the object detector network. It includes a total of 134.7 million parameters. It consists of 21 convolutional layers and 2 pooling layers. PANet network architecture has a VGG16 network as the backbone network and an FPN network as the object detector network. It includes a total of 14.7 million parameters. It consists of 31 convolutional layers and 5 pooling layers. Table 3 shows the introduction of the pre-trained model.

Table 3. The introduction of the pre-trained model.

Petrained Model	Convolutional layers(levels)	Pooling layers(level)	Parameters
U-net	23	4	7,759,521
Mask R-CNN	22	5	134.7M
PANet	31	5	14.7M

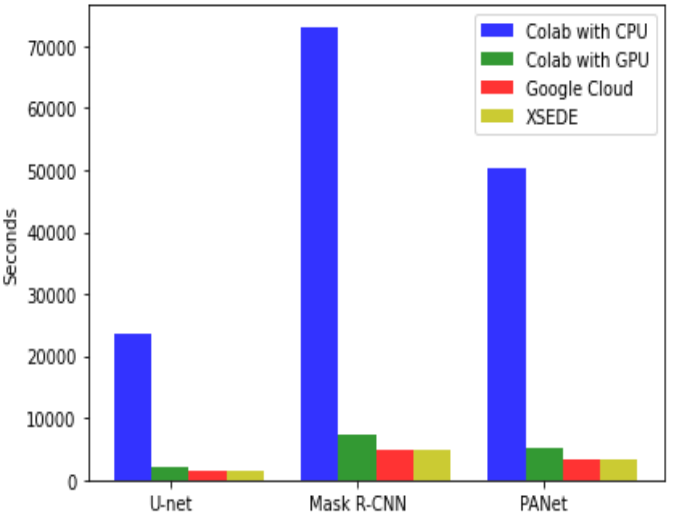


Fig. 15 The result of the training model in three architectures.

The experiment uses the three CNN architectures to train the model on three computing platforms respectively. The input image is RGB formats and the size of the image is 224 × 224. From the experiment, the accuracy of U-net is 57%, the accuracy of Mask R-CNN is 65%, and the accuracy of PANet is 63%. These accuracies are lower because the number of the dataset and the size of images are small. The result of the experiment shows U-net architecture spends the shortest time training the model because it has minimum parameters and convolutional layers. Mask R-CNN and PANet are both based on Fast R-CNN. Mask R-CNN spends the longest time because it has the most parameters than PANet networks. Besides, PANet includes an FPN network that is parallel. Therefore, PANet spends a shorter time than Mask R-CNN. Comparing four computing platforms, the efficiency of high-performance computer with GPU is lower than Google Colaboratory (Colab) with CPU and GPU. Compare Google

Cloud with HPC and XSEDE with HPC, they run with similar time because they have the same hardware resources. Fig. 15 shows the result of the training model in three architectures.

5. Conclusion and Future Work

Image classification, image detection, and image segmentation are the most basic applications in image processing. Robots and self-driving cars often use image processing to identify the presence, location, and type of one or more objects. So far, CNN network is a widely used technology to solve image processing problems. However, if the CNN network is applied to computer vision, the computing platform is a challenging problem. This paper tested and compared the performance of different computing platforms with several popular CNN network architectures in different image processing applications. Each CNN network architecture represents different trends that are developing in applications. The results of the experiment show we were able to successfully train our model in a high-performance environment. Because HPC owns more flexible hardware resources than Google Colab, HPC is more suitable for CNN network applications. In the future, we will use other HPCs on the cloud to test the performance of CNN network architectures.

Supporting information

Not Applicable.

Conflict of interest

There are no conflicts to declare.

References

[1] R. Pandey and N. Badal, Social Science Research Network, Rochester, NY, SSRN Scholarly, 2019, 3350311, doi: 10.2139/ssrn.3350311.

[2] Z. Chen , Y. Liu , Y. Wang , L. Zheng , M. Li , Y. Wang, 48th International Conference on Parallel Processing (ICCP), 2019, **99**, 1-10, doi: 10.1145/3337821.3337839.

[3] X. Zhang, F. Reveriano, J. Lu, X. Fu, and T. Zhang, IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2019, 40–42, doi: 10.1109/CSE/EUC.2019.00017.

[4] M. A. S. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, and R. Buyya, *ACM Comput. Surv.*, 2018, **51**, 1–29, doi: 10.1145/3150224.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, 1097–1105. doi: 10.1145/3065386

[6] K. Simonyan and A. Zisserman, 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015, 730-734, doi: 10.1109/ACPR.2015.7486599.

[7] C. Szegedy , W. Liu , Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, 1-9, doi: 10.1109/CVPR.2015.7298594..

[8] K. He, X. Zhang, S. Ren, and J. Sun, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 770-778, doi: 10.1109/CVPR.2016.90.

[9] X. Fu, J. Lu, X. Zhang, X. Yang, and I. Unwala, IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2019, 225–229, doi: 10.1109/CSE/EUC.2019.00050.

[10] F. Chollet, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 1800-1807, doi: 10.1109/CVPR.2017.195.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 779-788, doi: 10.1109/CVPR.2016.91.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. C. Berg, Computer Vision – ECCV 2016, 2016, **9905**, 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, IEEE Conference on Computer Vision and Pattern Recognition, 2014, 580-587, doi: 10.1109/CVPR.2014.81.

[14] S. Ren, K. He, R. Girshick, and J. Sun, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, **39**, 1137-1149, doi: 10.1109/TPAMI.2016.2577031.

[15] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767 [cs], Apr. 2018. doi: 10.4236/ce.2012.33059.

[16] J. Long, E. Shelhamer, and T. Darrell, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 39, 640-651, doi: 10.1109/TPAMI.2016.2572683.

[17] G. Sharma, D. Liu, E. Kalogerakis, S. Maji, S. Chaudhuri, and R. Mvech, Computer Vision – ECCV 2020, *Springer*, 12352, doi: 10.1007/978-3-030-58571-6_16.

[18] O. Ronneberger, P. Fischer, and T. Brox, Medical Image Computing and Computer-Assisted Intervention – MICCAI, Springer, 2015, **9351**, doi: 10.1007/978-3-319-24574-4_28.

[19] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 936-944, doi: 10.1109/CVPR.2017.106.

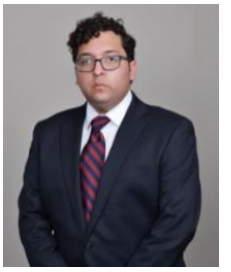
[20] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 6230-6239, doi: 10.1109/CVPR.2017.660.

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, **42**, 386-397, doi: 10.1109/TPAMI.2018.2844175.

[22] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, Computer Vision – ECCV, 2018, 11211, doi:10.1007/978-3-030-01234-2_49.
[23] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, 8759-8768, doi: 10.1109/CVPR.2018.00913
[24] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, 7151-7160, doi: 10.1109/CVPR.2018.00747
[25] L. Nwosu, H. Wang, J. Lu, I. Unwala, X. Yang, and T. Zhang, IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2017, 1318–1321, doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.213.
[26] S. Liu and W. Deng, 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015, 730–734, doi: 10.1109/ACPR.2015.7486599.
[27] K. He, X. Zhang, S. Ren, and J. Sun, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 770–778, doi: 10.1109/CVPR.2016.90.
[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 2818-2826, doi: 10.1109/CVPR.2016.308
[29] Z. Chen, K. Wu, Y. Li, M. Wang, and W. Li, *IEEE Access*, 2019, **7**, 80622–80632, doi: 10.1109/ACCESS.2019.2923016.
[30] H. Nguyen, *Math. Probl. Eng.*, 2019, **2019**, 3808064, doi: https://doi.org/10.1155/2019/3808064
[31] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, and R.-S. Jia, *IEEE Access*, 2019, **7**, 133529–133538, doi: 10.1109/ACCESS.2019.2941547.
[32] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, *IEEE Access*, 2018, **6**, 61677–61685, doi: 10.1109/ACCESS.2018.2874767.

Author information

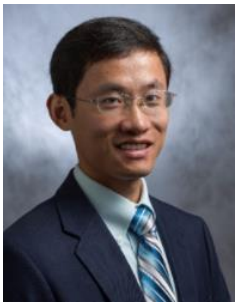
Xin Zhang *received his bachelor degree in Computer Science from University of Houston, Houston, TX, USA, in 2017. Now he is pursuing his master degree in Computer Engineering from University of Houston-Clear Lake, Houston, TX, USA. His research interests include image processing and deep learning.*



Francisco *is a Lead Quantitative Model Developer at USBank. He earned a Bachelor's in Economics from Grinnell College and Computer Engineering from the University of Houston-Clear Lake. During his Bachelor he conducted Deep Learning research at the Pittsburg SuperComputer Center and the National Center for SuperComputer Applications (NCSA). He then proceeded to complete his Master's in Computer Engineering at Duke University. There he conducted further research in Computer Vision for Duke's Applied Machine Learning Laboratory. Professionally he has focused on model development for Neocova, RetinalCare, and USBank.*

Dr. Ting Zhang *holds a Ph.D. in Electrical Engineering from the University of Alabama. Her main research interests include sensor systems, intelligent algorithms and virtual reality. She has published more than twenty papers on referred journals and conference proceedings, as well as a book. She teaches a wide range of courses in computer science, including computer organization, programming languages, data structures, and web programming. She supervises students on senior projects and research activities.*

Dr. Xingang Fu *received his Ph.D. degree in electrical engineering from the University of Alabama (UA), Tuscaloosa, AL, USA, in 2015. He is currently a Visiting Assistant Professor with the Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville (TAMUK), Kingsville, TX, USA. His current research interests include Neural Network, Deep Learning, Neural Dynamics, Approximate Dynamic Programming, Smart Inverter, Smart Grid, Wide-Bandgap Semiconductors (GaN &SiC), DSP/Microcontroller Embedded Systems, and so on. He is a member of the IEEE Computational Intelligence Society, IEEE Control Systems Society, IEEE Power & Energy Society, IEEE Young Professionals, etc.*



Jiang Lu *received his Ph.D. degree from the University of Alabama, Tuscaloosa, AL, USA, in 2015 in the field of Electrical and Computer Engineering. Before that he received the Master of Science degree from University of Florida, Gainesville, FL, USA and the Bachelor of Science degree from Shanghai Maritime University, Shanghai, China, both in Electrical and Computer Engineering. He is currently an assistant professor in the Department of Computer Engineering at University of Houston - Clear Lake, Houston, Texas, USA. His research interests are in the areas of intelligent sensor systems, Human-Machine Interaction, Cyber-Physical System, and Wireless Healthcare.*