International Journal of Scientific & Engineering Research, Volume 8, Issue ꞰȮ ×ÙĐÓı l ɣhυƛ
ISSN 2229-5518

Use of CNNs for the development of multimedia applications: A Virtual Assistance System

A Comparitive Study

Akshay BN, Muthu Santhiya S, Vibhu A Bharadwaj, Dr. Rajeshwari Hegde

Abstract – Computational needs of multimedia based applications are met by multicore platforms supplemented with associated algorithmic support. In the context of Internet of Things and cloud as a connecting infrastructure, multimedia requirements pose a higher order thread owing to the huge number of nodes and connectivity. A Convolutional Neural Network (CNN) based supplement to an IoT infrastructure is proposed as an extensible solution with assured scalability

Index Terms – Multimedia, CNN, Virtual Assistance, IoT, Caffe, Torch, Theano, TensorFlow, OpenCV

1 INTRODUCTION

Multimedia has become an indispensable part of everyday life and its processing and transmission is passing through stages of maturity in technology as multimedia demand is ever increasing with varied stringent requirements. What began as a slide show of images accompanied with an audio track in 1970s has made its debut into life critical applications such as training of medicos on 'virtual surgery'. While animation enabled multimedia is acclaimed to be one of best innovations in the recent past, very soon the corporate and research fraternity could, through multimedia processing, make users a part of a nonexistent world through the concept of virtual reality.

Multimedia content, typically, consists of combinations of text, images, animations, audio and video along with, possibly, one or more signals that manifest some sort of system dynamics in an abstract manner. Such content poses technical challenges with respect to dimensions that include storage, processing, transmission and presentation. Whereas the storage and transmission related technical challenges are alleviated through technologies such as compression/decompression techniques and feature based storage/retrieval techniques, transmission challenges are, specifically, addressed using relevant communication paradigms. This situation leaves one with challenges associated with multimedia processing most important of which is the inability to achieve real time processing of multimedia content. Multimedia processing application software based on highly optimized library support such as OpenCV working upon full fledge computing hardware, possibly, with multicore support shall be in a position to meet the real-time requirements of multimedia content. Recently, the multicore hardware has been replaced with GPGPU (General Purpose Graphic Processing Units) hardware innovated by Nvidia. Most of the multimedia based applications expose spatial parallelism and as such, ideally, real time multimedia processing with linear speed-up could be achieved on a generic parallel processing hardware architecture that employs either the multicore or GPGPU hardware as the fundamental building block. However, while the cost of building such architecture is, prohibitively, expensive, making the situation worse, the constraints on the real-time processing of complex multimedia processing applications continue to grow defeating the effectiveness of any of such paradigms. The main objective of this paper is to elaborate, by far, on the most complex application space in the backdrop of Internet of Things (IoT) and to propose a scalable solution thereof. This paper is organized as follows: Section II presents the complex application space and the associated challenges. Section III compares the various existing software platforms which are generic but have been successfully employed for complex image processing and have enough potential to address multimedia content processing in its true sense. These software platforms are based on Convolutional Neural Networks (CNN) which is explained in Section IV. Section V presents CNN based architecture for a class of such complex multimedia processing application space. Conclusions of the present work are done in Section VI.

2 ISSUES AND CHALLENGES

2.1 Dawn of IOT and Cloud

The need to process larger data faster is what drives innovation in computation. To this end, distributed computing has come a long way to help progressively build more and more complex frameworks to handle real-time multimedia requirements. Newer multi-core complex architectures lay the foundation for parallel and distributed applications to process the data.

The rise of the framework which led to the eventual development of the Internet of Things (IoT) demands a more suitable way to enable its users to process multimedia and its true nature of ubiquity will not be realized until it

International Journal of Scientific & Engineering Research, Volume 8, Issue ĸÕ ×ÙÐÓı lŸŀλ
ISSN 2229-5518

becomes more media friendly. The current research and development in IoT does not mandate the features of multimedia objects and this may prove to be a setback for IoT being embraced by the world in a truly complete and holistic fashion manner.

The success of content driven social media is because of its capability to process multimedia data. Users would not be attracted to a slow loading social media site. This is possible because of the use of cloud technology in handling data. Cloud technology provides users with large amounts of data storage and provides quick get and view services. While cloud technology is progressing at a rapid pace, where we lag is in the hardware architecture to support this technology. Cellular service businesses have recognized this concern and are upgrading their telecommunication infrastructure to cater to the use of multimedia driven services. However, cloud technology and required hardware infrastructure are technologically at different stages in their capabilities at the present stage with respect to the maturity of their technology.

Summarizing, the prime need of applications like those of social networks which has, intrinsically, been the ability to carry out large grain size computation has become multifold owing to multimedia enabled applications of which social networks is a significant one. Secondly, IoT which has adapted cloud as a means for connectivity, in years to come, would witness multimedia enablement as a mandatory requirement. Thus, it would be ideal to have both the cloud infrastructure and the IoT network infrastructure to be supported by a common paradigm which can operate in a broad spectrum of computational requirements starting from big grain size computation on cloud servers up to, relatively, smaller grain size of computation of the IoT network infrastructure. The essence of this paper is to present such an extensible paradigm grown into architecture to meet multiple applications' multimedia requirements while ensuring scalability.

The FIT-IoT (Future Interest Testbeds – IoT) Lab encourages users to use IoT technology and experiment using different WSN nodes. The Lab Node is a hardware architecture that is setup and made available to the user during experimentation. While FIT-IoT is open access and provides free storage to users, it will become increasingly hard to provide such facilities as more users embrace IoT for multimedia related uses. The aim of FIT-IoT Lab is to provide a scientific test bed for users to test their frameworks without the hassle of arranging for the necessary hardware. It emboldens developers without knowledge of hardware implementation to test their work. The quick deployment and result collection-analysis provides a very lucrative and attractive offer for users. The limitation of this however, is that it offers a platform to test small sensor networks for a specific set of use-cases. Even a robust test-bed solution like the FIT-IoT Lab has not been able to recognize the need for a more media focused approach when it comes to evolving the IoT technology. Similarly, GPU based hardware from Nvidia viz. the Nvidia Jetson TK1 series of hardware has the necessary resources to get multimedia enabled but is prohibitively expensive to get into an IoT space. Further, a gigantic effort is by IBM in building Watson IoT platform where, again, there is no big focus on multimedia requirement of future IoT node and networks.

2.2 Issues related to Standardization

While many organizations and people continue to work on IoT, which is as yet not a very mature technology with respect to standards, the OpenWSN tries hard to achieve its goal of creating an open-source hardware and software implementation platform for all users irrespective of the protocols which users may use in their work. This is both complex and challenging. However, here too metadata is given the spotlight while multimedia takes a backstage role. Standardization of multimedia protocols in IoT is in a nascent phase. The Open Source community welcomes both FIT-IoT Lab and WSN with open arms but fails to recognize that both these initiatives may find it hard to exist when multimedia takes the stage. They may not be able to provide for many users and they may even cease to be Open Source.

The incredible amount of research spending on artificial intelligence and machine learning has led to the development of platforms like the IBM-Watson which combines the best of computing and intelligence. However, even here, multimedia functions take a back seat even as IBM newly introduced Watson Natural Language Understanding API. IBM Bluemix, the cloud platform which can cater to different types of applications including multimedia related ones, has matured into a robust PaaS (Platform as a Service) but, the necessary hardware architecture required for development of platforms is absent. The API's on Watson's related use in multimedia applications are scarce and have great limitations of use.

Interestingly, even the Watson webpage points us to Deep Learning being the solution for collating and analyzing unstructured data. This suggests to us that there is a dire need of performance enhancement to be carried out on mature hardware platforms like the Raspberry Pi on Linux kernel or on ARM based hardware with Android OS that we see on smartphones, so that they can match up to the fast-paced and well-rounded evolution of IoT and related technologies. CNN's come to our aid here with their strong foundations in Deep Learning and their ability to process not only metadata but also multimedia data seamlessly.

International Journal of Scientific & Engineering Research, Volume 8, Issue
ISSN 2229-5518

3 Tools

Neural Networks and Deep Learning have found many applications from a simple photo search on Google to medical imaging. Hence, software that helps in implementation of neural networks was developed. Some of the described softwares are just libraries incorporating many function calls such as various forms of convolution, filtering and those which aid in our need to develop a novel CNN model. A few of the available tools are described.

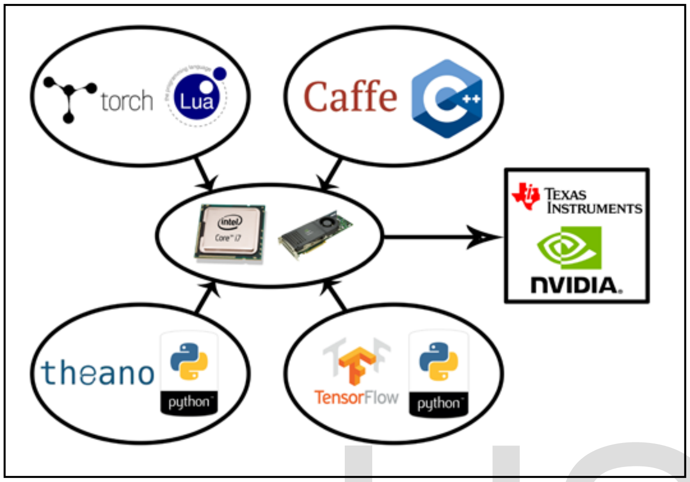

Fig 1: The Methodology to develop a CNN platform

3.1 Caffe, Torch, Theano, TensorFlow

Caffe: It is a deep learning framework. It has a Python based API that can be used for Machine Learning applications. Both CPU and GPU can be used by changing the status of a flag. Models can be trained without any code.

	Caffe	Torch	Theano	TensorFlow
Languages	C++, Python, MatLab	Lua, Python	Python	C++, Shell Scripts
Multi-CPU support	Yes	Yes	Partial	Inception
Multi-GPU support	No	No	Experimental	Yes
Readable Source Code	C++	Lua	No	No
Regional CNN support	Yes(best)	Yes	Mediocre	No

Table 1: CNN development libraries: A Comparison

Torch: It is a computational framework and the API is written in Lua. It provides fast and efficient GPU support. Graphs of Neural Networks can be built and parallelized over the CPU and GPU.

Theano: A Python library for mathematical computation based on multi-dimensional arrays. It provides accelerated performance using GPU. Theano compiled functions to work with NumPy are available. It is efficient when working with RNNs.

Tensorflow: It is a software library that uses data flow graphs for numerical computations. The nodes in the graph represent the mathematical operation and the edges represent the multidimensional array data that is passed on from one node to the other.

Tools	Description	Core Language	OS	Application
OpenCV	Image Processing toolkit	C, C++	Cross - Platform	Enable computers to retrieve information by processing images and videos
OpenGL	Environment to develop 2D and 3D graphics application	C	Cross - Platform	Hardware accelerated rendering
SceneLib	Library for Simultaneous Localization and Mapping (SLAM)	C++	Linux	MonoSLAM
ARToolKit	Open source tracking library	-	Cross - Platform	Augmented Reality
Unity	Game Engine	C, C++	Deployment across various OS	Provides 2D and 3D game development
CNNdroid	GPU accelerated library	-	Cross - Platform	Allows execution of CNNs on Android

Table 2: Comparitive study of different Image Processing Softwares

3.2 OpenCV, OpenGL, Scenelib, OpenScene Graph, AR Toolkit, Unity 3D

OpenCV: This is a traditional image processing tool kit. A brief description about it is given in the introduction section.

OpenGL: It provides an environment to develop 2D and 3D graphics application. These applications can be deployed over a wide range of platforms making in

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

portable. GLUT helps in portability by providing an API to execute OpenGL programs.

Scenelib: A C++ based library for Simultaneous Localization and Mapping and allows real time processing by interfacing a camera (MonoSLAM).

OpenScene Graph: It is a C++ and OpenGL based 3D graphics toolkit. It has high portability and is used in game development, virtual reality, visual simulation and visualization.

AR Toolkit: It is an open source tracking library that helps developers build Augmented Reality applications

Unity3D: It is a game engine that provides 2D and 3D game development. These games can be deployed across devices. Unity 3D and OpenScene Graph are supported and coding can be done using various languages.

4 Convolutional Neural Networks

Convolutional Neural Networks are seen to be the spearheads leading to exploration and experimentation of Deep Learning concepts to help solve problems that involve processing large meta and multimedia data. The data can be unstructured, unclassified and even random. The ingeniousness of CNNs lies in its ability to build a platform to process multimedia by enhancing the capability of hardware frameworks using simple convolutional layers. Such capabilities advocate the adoption of evolving technologies like IoT in a well-rounded manner.

5 Solution Phase

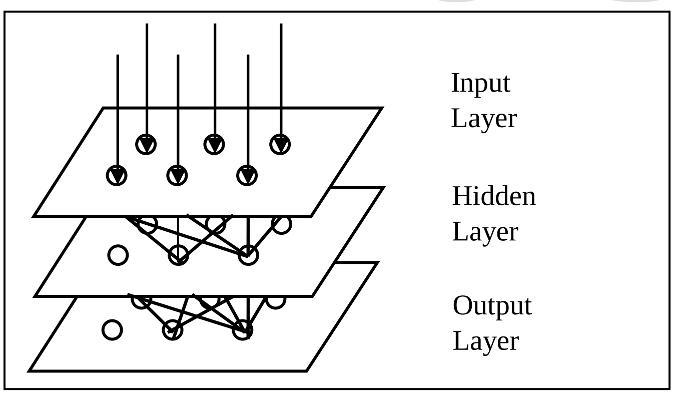

Fig 2: CNN development Layers

Convolutional Neural Networks (CNNs) can be considered as a special architecture of neural networks that is specifically used for image recognition. CNNs convolve the input given using a filter of a specific size applied over the entire input patch. These make them sensitive to or learn a certain feature or structure of the input. Detection models based on Convolutional Neural Networks are performance efficient and highly accurate due to the training of the neural network widely using datasets. They are used from applications such as Facebook for its automated photo tagging to self- drive cars as they can be used to identify faces, traffic signals or other objects such as cars.

When an image is given as an input the system, the computer sees it as an array of pixels. While the human brain can effectively identify, the details present in the image after viewing it through eyes, the neural network does so by identifying the low-level features such as edges and corners and then using the information learnt during the training phase to classify the image.

At the crux of Convolutional Neural Networks is the process of convolution. Convolution is, simply, a sliding window function applied to its input layer (a matrix of numbers) and computing the corresponding output layer. The matrix of numbers could be an image data where each number signifies a pixel. The sliding window is called a kernel, filter or a feature detector. Each layer of CNN (corresponding to input layer or the hidden layers of an ANN), typically, applies hundreds of kernels and combine their results. During the training phase, each CNN layer, automatically, learns the values of each of its kernels. Secondly, unlike the case of Artificial Neural Nets (ANN), each neuron of a specific layer is not connected to all neurons of the succeeding layer. The inputs to a layer is apportioned among different kernels of that layer. Applying this paradigm to an image classification problem, a CNN layer may learn to construct edges from raw pixels in the first layer, learn to construct small shapes in the second layer and eventually at some layer may learn to recognize a specific object. None of the connectivity between successive layers is fully-connected. However, the final layer is the Fully Connected Layer, which is used for the classification of the image. The program consists of a number of classifications that a feature of the image might fit into. This layer considers the output of the previous layer as an input and detects the closest class that the features in the image correspond to and makes predictions.

The training phase of a neural network involves iterative of a relevant error minimization algorithm similar to back propagation algorithm. During the first iteration, the program is unable to extract any low-level features or classify the image. A loss function is used to calculate the error factor. This error is used to alter the kernel values and the resulting connections so as to minimize the error. Once acceptable error is reached, the network is said to be trained and usable for testing.

The inconvenience with respect to CNN is that the cost involved in the training of the neural network and the time taken to arrive at an output is high. This has forced the need for an acceleration mechanism. Different acceleration mechanisms have been proposed such as the convolution operation in the spatial domain being converted to dot operation in the frequency domain [6], CNNdroid [7] which is a set of library functions that helps in GPU acceleration of a CNN code and run it on Android, parallelization of tasks on GPU or implementing Fourier Transform using Open

International Journal of Scientific & Engineering Research, Volume 8, Issue
ISSN 2229-5518

Computing Language (Open CL) on the convolution based detection model [8].

Thus, an accelerated neural network is necessary for the real-time processing of multimedia data.

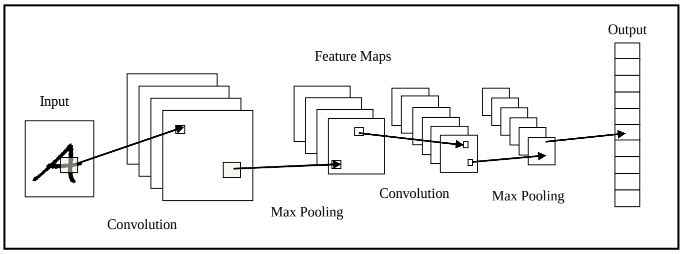

Fig 3: CNN methodology

6 Platform for Experimentation

In the back drop of the gamut of tools briefed earlier and their support to build CNN model alongside the critical issue associated with multimedia applications with emphasis on IoT and cloud based applications, the need of the hour is to build an experimental test bed for research investigations on CNNs for their applicability. Such an experimental test bed as depicted in Fig.1 is built by authors and employed to execute case studies briefed in the following section. The test bed hardware, simply, consists of a host computer that runs Ubuntu 16.04 which is interfaced with an embedded system. The host computer is equipped to receive the image/video data either from open source repository or could even capture the image and create the repository. Basically, the CNN augmented processing software operates on the data and trains the network for the intended mission. Once trained, the weights associated with the kernel and the CNN model itself is downloaded into the embedded system. Thus, the host computer assumes the resource intensive training activity whereas the embedded system executes the testing phase or the real-life execution of the intended objective. The software architecture of the test bed is depicted in Fig.1. The target deployment port is customization software that maps the API calls usable on the host computer with its embedded counterpart. The TDP shall have to be ported whenever the embedded hardware changes. In the present work, a smart phone running Android 4.0 is used as the target hardware.

7 Usecases

The very first application we aimed to develop involved the use of the CIFAR-10 dataset consists of 60000 color images in 10 classes with 6000 images per class. The size of these images for versatility was 32 pixels by 32 pixels. The data set consisted of 50000 training images and 10000 test images. (put appropriate reference in this)

The dataset is first divided into five training batches and one test batch with each consisting of 10000 images each. The training batch and the test batch differ in, not only the number of images they have to offer, but also in the number of images from each class they input to the convolutional layer of the neural net. The test batch contains exactly 1000 randomly-selected images from each class. The training batches however may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The model CIFAR-10 chosen was tested using an above-mentioned tool called as Torch. Mean Accuracy levels in running and testing the model on various different data sets are compiled and tabulated in table below.

Iteration Number:	Accuracy level after 10 epochs:	Mean accuracy value:
1.	98.425	-
2.	97.415	97.920
3.	98.620	98.153

Table 3: Output comparison

Another implementable idea which we took to working on was based on thnets. Thnets is the work of Marko Vitez who has compiled together a basic library which can run neural networks created using the aforementioned software tool called as Torch. The idea behind having a compiled library includes the versatility with which we may be able to import developed neural networks (some which have that have been developed from scratch and some which have been obtained from biological neural structures) directly into the test code and run it. One of the main requirements which were stated for the successful implementation of thnets was OpenBLAS (Open Source – Basic Linear Algebra Subprograms). This OpenBLAS is an API with hand crafted optimizations for several processor architectures like Intel's Sandy Bridge and Loongson. Successful implementation of the thnets module for Image/Object recognition and classification gives us a forward path time of about 7 seconds for the direct implementation without the use of Hardware Accelerators such as cuDNN (Nvidia CUDA Deep Neural Network library).

The final and holistic environment we built includes an Android application which could make use of the host phone's camera to take images for Object Recognition and Classification. This was accomplished through the use of a trained Caffe model. This specific application makes use of Fast R-CNN for Object detection. One of the major things to keep in mind while considering this specific model of Fast R-CNN is the concept of R-CNN which is Region-based Convolutional Neural Network. Fast R-CNN was created by Ross Girshick at Microsoft Research, Redmond, USA. It is a framework for faster object recognition when compared with regular neural networks. Fast R-CNN runs almost 200

International Journal of Scientific & Engineering Research, Volume 8, Issue
ISSN 2229-5518

times faster than regular R-CNN and almost 10 times faster than Spatial Pyramid Pooling in a test real-time scenario.

The developed Android App on the basis of this concept of Fast R-CNN includes a license from MIT open-source image datasets. It has a support for Android version 4.0 and upwards and supports all devices which have a 64-bit architecture and/or an ARM v7 processor. The Caffe model and weights used for proper functioning of the CNN network is pushed onto an SD card and is input to the Phone's SD card slot. The entire system is built using Gradle on Ubuntu or on Android Studio on Windows before exporting the. apk file to the phone.

The developed Caffe model is capable of both object as well as scene detection. An example output is shown below.

Fig 4: Sample Output

Although in this section we have spoken about multiple different applications and methods which we have used for development of Object Recognition and Classification, the main intention behind such a vast study lies in the scope for future IoT systems. The main idea here is to come out with a holistic IoT network which is, by itself, a separate and autonomous entity. Present day IoT nodes act as channel gateways for transfer of information from one device to another. No amount of machine intelligence and autonomy exists in their part and are thus unable to work more efficiently and achieve a more well-rounded connectivity within the nodal network. By developing support structure to improve these nodal networks, as described below, we may be able to come out with a new age of IoT nodes. The idea is to come out with a system architecture which consists of a Raspberry Pi or a cellphone running Android which is capable of inter/intra communication within a given nodal set in a more effective way than present day IoT nodes. Such smart IoT networks warrant multimedia as a critical resource whose very presence increases the computational burden wherein the solution framework based on CNNs would become handy.

8 Conclusions

CNN based computational support for multimedia enabled applications with emphasis on IoT and Cloud platform is presented in this paper. Use cases employed elsewhere have been experimented using software tools briefed in section III.

9 Bibliography

[1] C. Dubout and F. Fleuret, 'Exact acceleration of linear object detecotrs in Computer Vision' -ECCV 2012, pp.301-311, Springer, 2012

[2] Seyyed Salar Latifi Oskouei, Hossein Golestani, Matin Hashemi, Soheil Ghiasi, 'CNNdroid: GPU-Accelerated Execution of Trained Deep Convolutional Neural Networks on Android', 2016

[3] Qi Liu, Zi Ruang, and Fuqiao Ru, 'Accelerating Convolution-based Detection Model on GPU', 2015

International Journal of Scientific & Engineering Research, Volume 8, Issue ƘȮ ×ÙĐÓılɣɦɅ
ISSN 2229-5518