DOI: http://dx.doi.org/10.26483/ijarcs.v8i8.4613
ISSN No. 0976-5697
Volume 8, No. 8, September-October 2017
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

MACHINE LEARNING FORENSICS:A NEW BRANCH OF DIGITAL FORENSICS

Prerak Bhatt
IFS, Gujarat Forensic Sciences University,
Gandhinagar, Gujarat

Parag H. Rughani, Ph. D.
IFS, Gujarat Forensic Sciences University,
Gandhinagar, Gujarat

***Abstract*—** The objective of this research is to understand how machine learning can be used in digital crime and its forensic importance, setting up an environment to train artificial neural networks and investigate as well as analyze them to find artefacts that can be helpful in forensic investigation.

***Keywords*-** Machine Learning, ML forensics, AI forensics, TensorFlow, AI related crimes

I. INTRODUCTION

Artificial Intelligence (AI) and Machine Learning (ML) has been around since a long time but it is now that we have enough computational power to effectively develop strong artificial neural networks (ANN) in a sensible time frame with the help of strong hardware and software support.

Companies like Google, Amazon, Samsung etc. are heavily investing in AI technologies and funding the research. Google CEO Sundar Pichai announced the company's vision to be "AI-First" at Google I/O 2017 and quoted "It's all about a transition, from searching and organizing the world's information to AI and machine learning." [1].

Google also unveiled "TensorFlow Research Cloud" program which will provide researchers with access to 1000 cloud TPUs (Tensor Processing Units) for free with a condition to open source their code and research. [2]

Since AI is becoming widely available to more and more people, the potential of the technology to be used for malicious purposes also increases significantly. To counter this and be prepared for forensic challenges regarding crimes committed with AI or ML, forensic evaluation and analysis of the technology is necessary.

This paper demonstrates implementation of a machine learning open source program "DeepQA" and forensic analysis of the same while the program was in training and testing modes. This paper also lists out some important artifacts findings that can be taken as a reference for cases in future to prove or determine that a machine learning technique based on TensorFlow was used on provided evidence.

II. FORENSIC IMPORTANCE

AI and ML has great advantages and holds a bright future ahead. But the same technology can inevitably be used to craft, automate and execute some serious crimes that can also be deadly for people.

For instance, hackers can develop an ANN that scans new versions of popular apps for unknown vulnerabilities, exploit them and/or report the vulnerability to the hacker. If this process is done manually, it can take long time to pentest an app. But with the help of ML it can be done really quickly and can be done on multiple different apps at the same time with machine efficiency. It makes the job for hacker really easy and quick.

Hackers are available to rent on dark web. It is a possibility now that AI powered bots will replace them and do the job more efficiently and quickly than a human being can. Earning more money to hacker than he did before.

The task in 2016 DARPA (Defense Advanced Research Projects Agency) Cyber Grand Challenge was to create an AI that can correct the provided buggy code itself, patch vulnerabilities present in its own system and look for intrusions by opponents with minimal human interaction. The winning prize was $2million. The competition lasted for about 8 hours. [3]

"Spear phishing is going to become really, really good when machine learning is incorporated into it on the attacking side," says Dave Palmer, director of technology at Darktrace, a cybersecurity firm which deploys machine learning in its technology. [4]

So, there are so many possibilities where ML and AI can make a criminal, hackers or a terrorist's job easy and quicker.

And this is why, ML and AI holds great forensic importance. It is a new field to dive in for forensic investigators and the scope of research is really big. There is no research found on the algorithms or frameworks of ML that suggest how to investigate or identify if any AI or ML technology was used in commitment of a digital crime.

III. ENVIRONMENT SETUP

We need a specific environment setup in order to develop ANN based programs. It requires a powerful CPU and/or GPU because training of an ANN model is a resource consuming task.

A. *Hardware*

- **CPU**: Intel Core – i5 6600k
- **GPU**: Nvidia GeForce GTX 1070
- **RAM**: 8GB

B. *Software*

- **OS**: Ubuntu 16.04

- **Parallel Computing Platform**: Nvidia CUDA
- **ML Library/Framework**: cuDNN & TensorFlow 1.0.0
- **Language Platform**: Python 2 & Python 3
- **ML Program**: DeepQA [5]
- **Dataset**: Cornell Movie Dialogs [6]

And other dependencies for above mentioned programs.

C. Why use GPU?

Training deep neural networks can be a time-consuming process. It involves a big amount of matrix multiplications and other mathematical operations that if parallelized, can boost up the calculation time significantly.

A single workstation CPU in current scenario might have 8-10 cores, while a single GPU can have thousands of cores. Although the GPU cores are slower than CPU cores, the large number of cores makes that redundant.

Nvidia GeForce GTX 1070 GPU has 1920 CUDA cores clocked at 1506MHz with 8GB of VRAM.

D. Why TensorFlow?

Google open sourced TensorFlow on November 9, 2015. Since then, it is the most sophisticated and well written library for Machine Learning. It supports CPU and GPU acceleration.

Because of its open source nature, it has attracted a lot of machine learning programmers and they are using it in various creative ways to build different types of programs and services.

Now after the announcements at Google I/O, the use of TensorFlow will spread more between developers and more open source projects will be pushed out.

Because of these reasons, I have chosen TensorFlow library to build a ML ANN.

IV. IMPLEMENTATION

A. Ubuntu 16.04 & nVidia utilities:

I installed Ubuntu 16.04 with a separate /home, / (root) and SWAP partitions on a workstation. Installed all necessary programs such as Python 2 and Python 3 and other dependencies that are required.

Installed latest nVidia graphics drivers, as well as installed cuda and cudNN utilities that provide tensorflow a bridge between Python and GPU for training a neural network.

B. Tensorflow:

Cloned tensorflow from its GitHub repository.

Configured tensorflow installation by running ./configure script in tensorflow directory. Configuration included specification of location of cuda installation, cudnn installation, compute capability of your GPU, python installation directory etc.

After configuring tensorflow correctly, I built a pip python package and installed tensorflow as a plugin on my python 3 installation.

C. DeepQA ChatBot Program:

DeepQA is an open source Neural Conversational Model. It uses a RNN (seq2seq model) for sentence predictions. It is based on Python and TensorFlow. The advantage of this program is that it supports a various number of conversational datasets available for research purpose.

DeepQA also provides code to setup a Django web server that gives the chatbot a nice graphical interface to play with.

The developer of this program is very responsive to queries and keeps the project up to date.

Because of this program's versatility and activeness, I chose this program to train an ANN chat bot.

D. Training chat bot:

To start training a neural network on "Cornell movie dialogs" dataset, I entered the following command:

$python3 main.py --corpus cornell

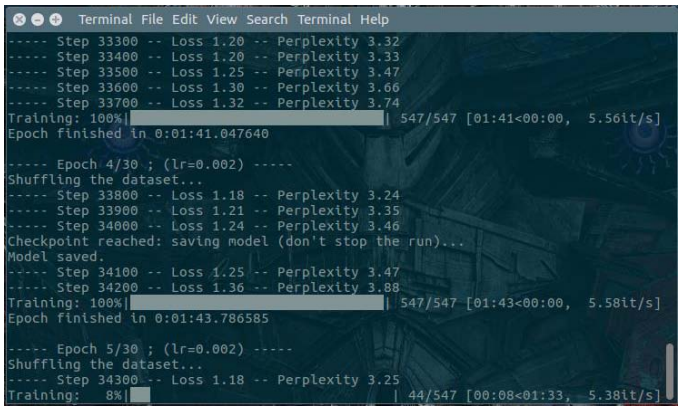

Figure 1: Training on

There are many different parameters and variables are being displayed in figure 1. Explanation of them is as following:

An epoch usually means one iteration over all of the training data. For instance, if you have 20,000 images and a batch size of 100 then the epoch should contain 20,000 / 100 = 200 steps.

The loss measure error between two tensors, or between a tensor and zero. These can be used for measuring accuracy of a network in a regression task or for regularization purposes.

Perplexity metric in ML is a way to capture the degree of 'uncertainty' a model has in predicting (assigning probabilities to) some text. It is related to Shannon's Entropy. Lower the entropy (uncertainty), lower the perplexity.

You can finish the training any time by pressing ctrl + c in the terminal. It will save the model as a .ckpt file in /save directory and exit the program.

You can resume training from the same step it left off later on if you want to.

Training an ANN generally takes a good amount of time depending on the size of provided dataset and how many epochs (one full training cycle) you are running your program for. Also the parameters you use in order to train the ANN effects the training time.

It took me about 7 to 8 hours to train an ANN model based on default parameters with 30 epochs and used "Cornell Movie Dialogs" for dataset. Initially, it did not give me good results.

But by trying and setting different values in parameters like learning rate (lr), max sentence length, etc. started to get me better results.

I spent more than 24 hours in total to train different models to get better results. Each try consisted of 30 epochs.

E. Testing chat bot:

To test the trained model and see how well has it trained based on our dataset, I entered the following command:

$python3 main.py --test interactive

It provides a command line interface where you can type a question or a message and the chatbot will reply to it.

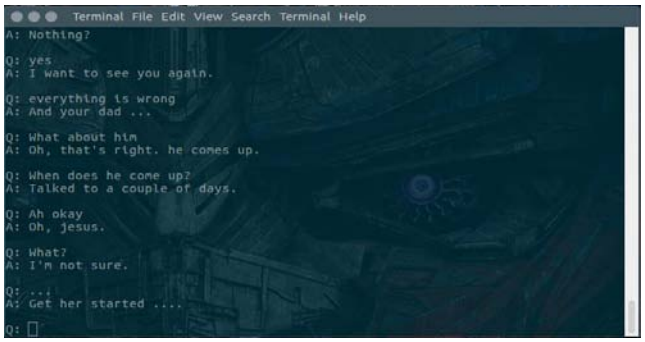

Figure 2 – testing the chatbot

The replies the bot is giving to questions in figure above. They are not really great but they are somewhat contextual based on the questions asked to it.

Training it on a better dataset and for longer timing with proper learning rate and other parameters can give you better results.

Now imagine we provide this model a dataset that consists of conversations between a support employee of bank and a client. If we train it properly and long enough then it will be able to successfully make the client believe that he is talking to a real legit person and he would trust him enough to reveal his information to him.

You can visualize the computational graph, the cost of the ANN and word embeddings for our model with TensorBoard, just run tensorboard --logdir save/ command.

Word embedding is the collective name for a set of language modelling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary size. [7]

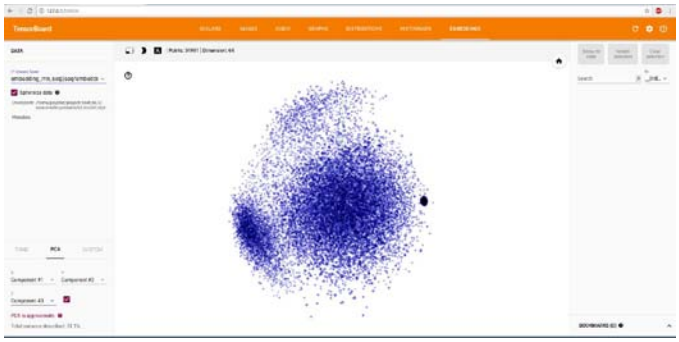

Figure 3 – TensorBoard word embeddings

The embeddings of our trained model can be seen in the screenshot above. It is pretty dense. Means it is a well-trained model containing a big amount of word vectors.

V. FORENSIC ACQUISITION AND ANALYSIS

After training a functional neural network that can give out decent output, its time to forensically analyze the system to find artefacts that help us determine that the system was used in generation and testing of a neural network based on TensorFlow.

A. Tools used:

- **LiME: "**Linux memory extractor" (LiME) is used to take live RAM dumps in .lime and .raw formats. [8]
- **Rufus**: To create a bootable Ubuntu 16.04 USB thumbdrive. [9]
- **Disks Utility:** It is a part of Ubuntu live system that lets you create images of different partitions or whole disk. [10]
- **EnCase:** EnCase is used to investigate disk images and RAM dumps to find relevant artefacts and to make a report based on findings and other technical information about the system. [11]

B. Live memory capture with LiME:

LiME is a Loadable Kernel Module (LKM) which allows for volatile memory acquisition from Linux and Linux-based devices.

LiME utilizes the insmod command to load the module, passing required arguments for its execution.

After cloning the source code of LiME from GitHub, it is needed to make a kernel module compatible with your Linux kernel. You cannot load a kernel module that is made for another kernel on your kernel. It can be fatal in some cases for the OS.

I loaded the LiME kernel module in the kernel while the DeepQA program was in training mode.

Figure 4 – LiME while training

After taking the RAM dump while program was in training mode, I put program in testing mode and again took a RAM dump following the same way.

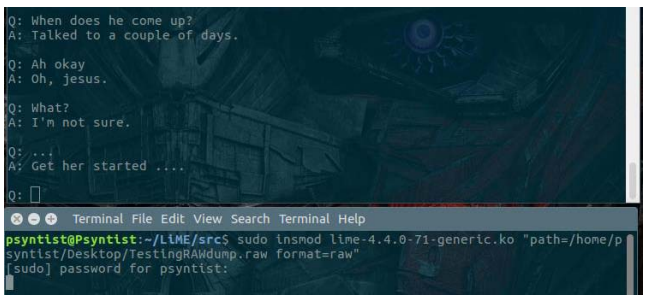

Figure 5 – LiME while testing

So now we have two different RAM dumps. One while the system was in training mode and one while the system was in testing mode.

1. TrainingRAMdump
2. TestingRAMdump

C. *Disk acquisition with 'Disks' Utility:*

"Disks" is a tool that comes preinstalled with Ubuntu 16.04. It lets you manage your hard disk partitions. You can create new partitions, edit partitions, shrink, extend, mount, unmount and take logical images of partitions in .img format.

I created an Ubuntu 16.04 live bootable USB thumb drive and booted it up on my system.

I launched Disks utility and selected /home partition. Clicked on settings icon on left and selected 'create logical image' of the partition and provided the location to store a bit-by-bit image of /home partition.

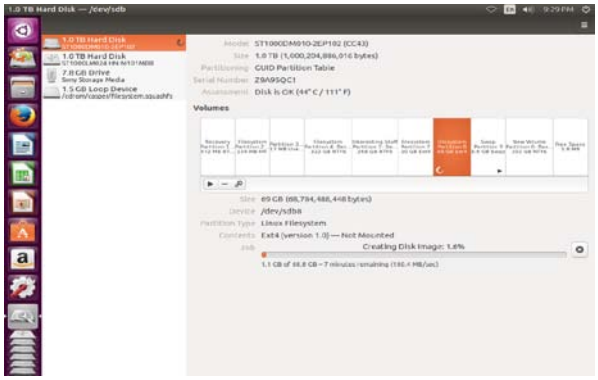

Figure 6 - /home image

I did the same procedure with the remaining partitions respectively, / (root) and swap.

So now we have logical images of all three partitions used on Ubuntu that will be loaded on EnCase for investigation.

The reason to acquire these partitions is explained below:

/home: DeepQA program is hosted on this partition as well as RAM dumps I took with LiME are stored here.

/ (root): Installation of TensorFlow and other dependency programs were done in this partition. Plus this directory is the parent of all directories on Ubuntu.

SWAP: Swap is used for paging. So it might have some volatile data stored that might be useful for the investigation.

D. *Forensic Analysis on EnCase:*

Now comes the most interesting part of this project. Analyzing the RAM dumps and hard disk images to find relevant artefacts.

I chose EnCase to analyse the evidence because EnCase provides state of the art solutions for evidence analysing, processing and report generating. The interface is also easy to use and clean.

Biggest advantage to use EnCase is that it can be cited in court of law in USA, India and other major countries.

I created a new case on EnCase and entered appropriate information such as the name of case, case number, examiner namem case ID etc.

After creating the case, I added evidence files one by one. First off, I started with adding the logical image of /home partition. After adding the image of /home partition as an evidence file, it is time to acquire the same evidence. EnCase makes .E01 image of the raw image of the evidence we provided in acquiring phase.

After acquiring the evidence image, I put the acquired evidence image on processing. Selected appropriate processing options like System Info Parser, File Carver, Personal Information extractor, Linux Artefact Parser, etc.

I followed the same procedure for acquiring and processing for next two logical images, / (root) and SWAP.

Processing SWAP partition did not give any categorized data it was shown as unallocated space but, some RAW data can be found from that unallocated space.

After adding, acquiring and processing all evidence files EnCase Evidence window looks like this:

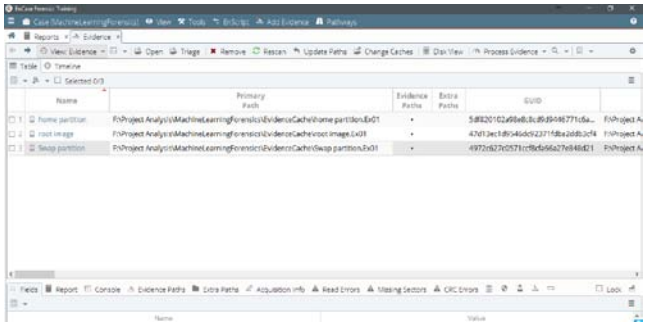

Figure 7 – All evidence images

E. *Findings:*

I started the analysis of evidence and found some concrete artefacts explained below:

Tensorflow Installation location:

One of the most primary and important artefact is to find out if TensorFlow is installed on the system.

TensorFlow is a Python library. So, first check the location of Python installation and then look for tensorflow inside it.

On any Linux based OS programs are installed under /usr directory so it is the first directory one should consider to analyze for Python installation.

Tensorflow is installed under /usr/local/lib/python3.5/dist-packages/tensorflow directory.

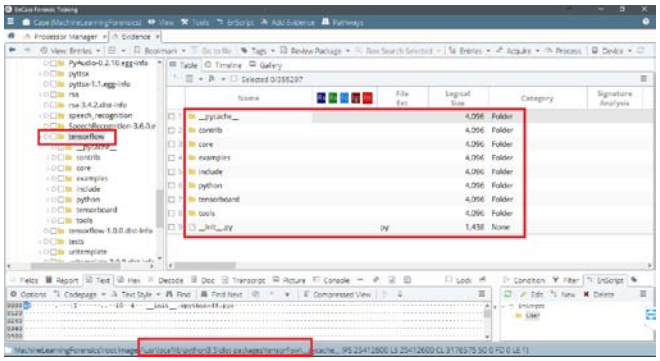

Figure 8 – TensorFlow location

Interestingly, this directory also contains some interesting Python libraries that can be used as a part of a ML program. Such as speech_recognition, pyttsx, etc.

Searching for keywords:

Seq2seq keyword:

seq2seq (sequence to sequence) is a class of tensorflow that is used in developing a sequence to sequence, RNN (recurrent neural network) model.

DeepQA program is based on seq2seq modelling and is a recurrent neural network. So the possibility to find this class used in creation of the model is high.

Netflix keyword:

Word Netflix was a part of our dataset I used to train our ANN. So using this as a keyword to search to see if we can find it in RAM dumps or on SWAP partition.

GTX1070:

If training of ANN program was done with tensorflow and GPU, it will include the name of the GPU used in the training at least somewhere in volatile data or in parameters of tensorflow.

Added gtx1070 as a keyword to see if we find some artefacts related to it as I used gtx 1070 GPU to train the chatbot.

Keyword hits:

It takes a good amount of time to analyze all evidence files for the provided keywords to EnCase. But it checks all evidence files thoroughly and even shows if keyword hit was found in unallocated space.

After the processing of searching for keywords finishes, EnCase shows you all the keyword hits in one window of Keywords. It shows the number of files and number of hits the keyword has got right next to the name of keywords. It can be seen in the screenshot below.

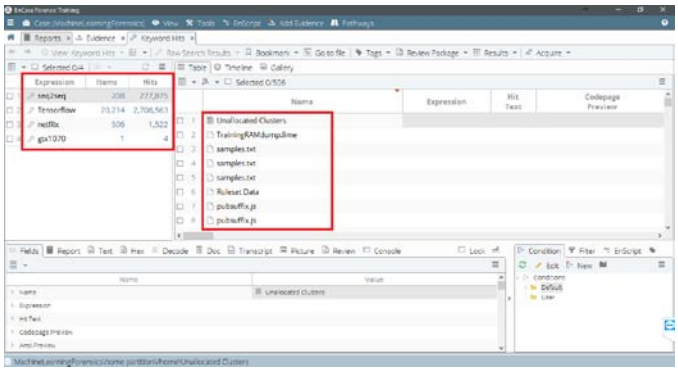

Figure 9 – Keyword hits

Seq2seq keyword hit:

Surprisingly seq2seq keyword got 277,875 number of hits in all evidence images. Meaning it has been used a lot in 208 number of files. I analysed some of those files and found following results.

Found the python script file of DeepQA chatbot.py containing the seq2seq keyword. It can be seen that tensorflow class seq2seq class is used in the code of this file.

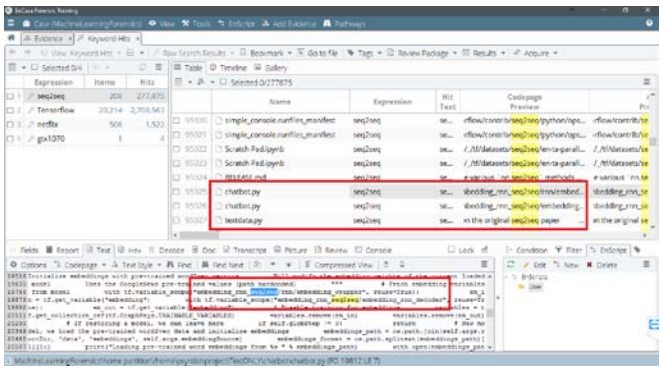

Figure 10 – seq2seq hit

The same keyword was also found in the compiled chatbot file chatbot.pyc it confirms that the script was indeed ran at least once on the system. The pyc (Python compiled) file only generates once the program executes at least once.

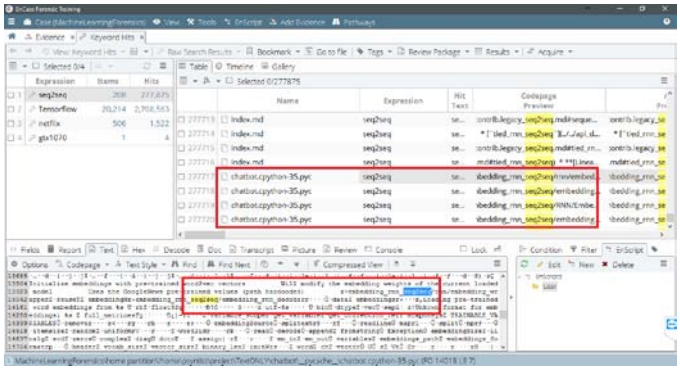

Figure 11 – seq2seq pyc

The keyword seq2seq was also found in the model.ckpt file of our chatbot. This also confirms that the training of an ANN was committed. Since we know that model file only generates once you start training a neural network.

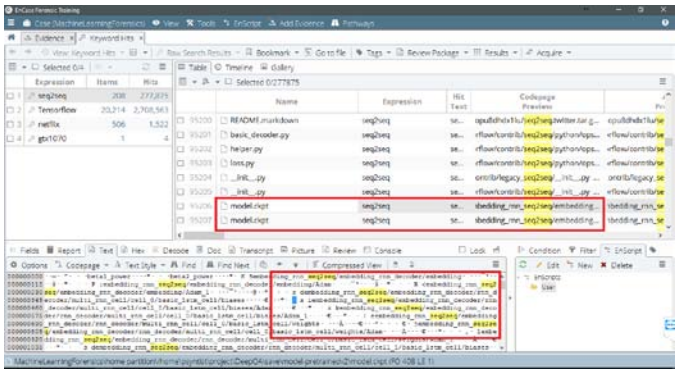

Figure – 12 seq2seq .ckpt

Netflix keyword hit:

I found Netflix keyword hit in some dataset files (.tsv). We can see that the word has been mentioned in a conversation between two parties in the file content.

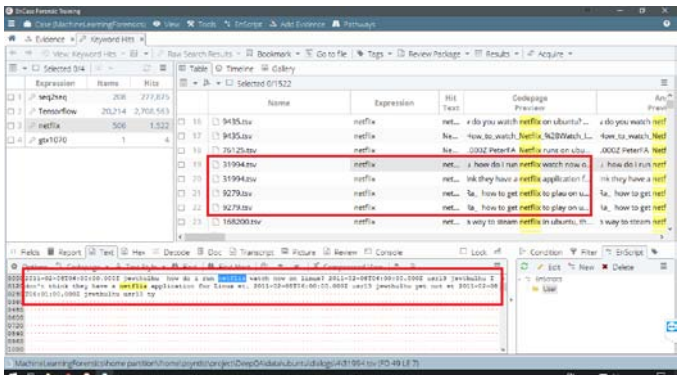

Figure 13 – Netflix dataset

found Netflix keyword in a dataset.pkl file. The pickle module (.pkl) implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. Tensorflow uses .pkl files when the program is in testing mode to give quick serialized outputs.

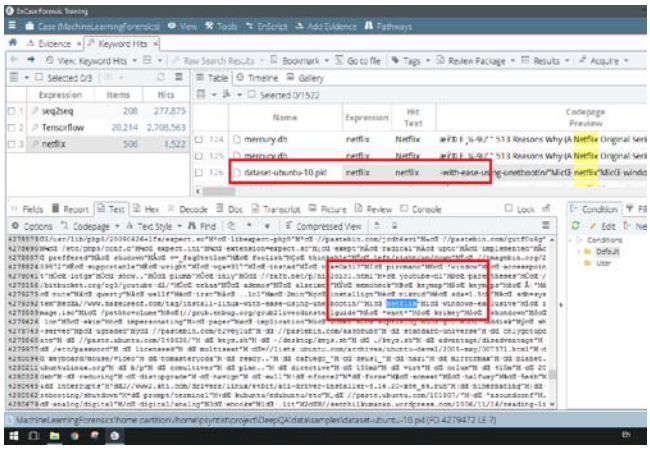

Figure 14 – Netflix pkl

One interesting find for this keyword was in TrainingRAMdump file that was captured while program was training. This artefact confirms that the dataset that contained this keyword was also used in training the program.

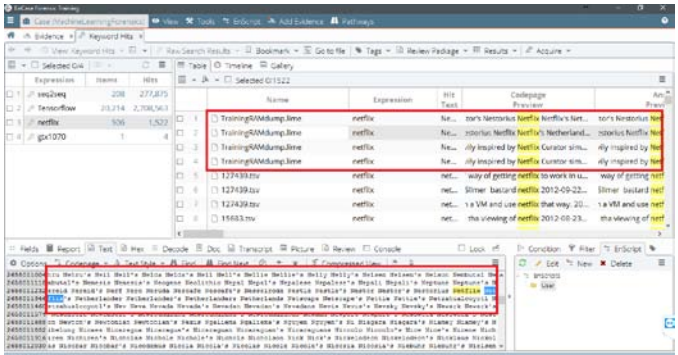

Figure 15 – Netflix RAM

Based on the artefacts I found regarding Netflix keyword, I can confirm that string 'netflix' was a part of dataset and the dataset was used while the program was training.

Gtx1070 keyword hit:

The keyword GTX1070 was found on SWAP partition. Since the SWAP partition is considered as unused disk area, EnCase shows it as one single raw file.

SWAP file is used for paging. So since the keyword is mentioned here along with strings like tensorflow in the content, we can conclude that GPU acceleration was used to train neural networks using tensorflow.

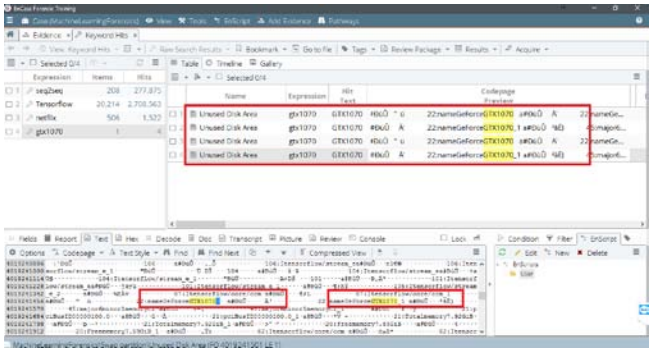

Figure 16 – gtx1070 in SWAP

VI. CONCLUSION

Building an ANN with the help of machine learning has got better with the introduction of Google's open source machine learning library TensorFlow.

After finding the relevant artefacts in the investigation of the evidence images, I can conclude that a machine learning program based on tensorflow was trained as well as performed on the system.

These findings can be used as a reference in future cases to detect or identify the use of machine learning libraries, algorithms, techniques etc.

However, a lot of research work still needs to be done in this field. Proper and deeper analysis of volatile information would be beneficial as well as more in-depth analysis of neural networks might help us to get more familiar with machine learning programs in the scope of digital forensic.

VII. REFERENCES

[1] https://venturebeat.com/2017/05/18/ai-weekly-google-shifts-from-mobile-first-to-ai-first-world/

[2] https://techcrunch.com/2017/05/17/the-tensorflow-research-cloud-program-gives-the-latest-cloud-tpus-to-scientists/

[3] https://techcrunch.com/2016/08/05/carnegie-mellons-mayhem-ai-takes-home-2-million-from-darpas-cyber-grand-challenge/

[4] http://www.zdnet.com/article/how-ai-powered-cyberattacks-will-make-fighting-hackers-even-harder/

[5] https://github.com/Conchylicultor/DeepQA

[6] http://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

[7] https://en.wikipedia.org/wiki/Word_embedding

[8] https://github.com/504ensicsLabs/LiME

[9] https://rufus.akeo.ie/

[10] https://apps.ubuntu.com/cat/applications/precise/gnome-disk-utility/

[11] https://www.guidancesoftware.com/encase-forensic