Machine Learning – Part 2
Written for Rootstrap, Inc.

By Ravi Das

Introduction

Our last whitepaper provided a detailed overview into what Machine Learning is all about. It is a component of Artificial Intelligence, and when these two are used together, they provide a very powerful toolset in terms of automation and predicting future threat variants, from the standpoint of Cybersecurity. More specifically, it covered the following topics:

- A technical definition of Machine Learning (ML);
- The key differences between ML and Artificial Intelligence (AI);
- Some of the present-day applications of ML which include the following:

  *Predictive Maintenance & Quality Control;

  *The recruiting of job candidates to help fulfill various job openings;

  *Providing a rich customer experience, especially as it relates to Virtual Agents and other forms of Chatbots;

  *Finance, especially when it comes to the Cryptocurrencies, such as the Bitcoin and virtual trading using the Blockchain.

- How an ML system learns, following these steps:

  *Data Order;

  *The picking of the appropriate algorithm;

  *Training the ML model;

  *Fine Tuning it based upon the preliminary results received.

- The differences between Supervised and Unsupervised Learning;

- The actual algorithms that are used today in ML which include the following:

  *The Naïve Bayes Classifier;

  *The K-Nearest Neighbor;

  *The Linear Regression;

  *The Decision Tree;

  *The Ensemble Model.

The second half of the whitepaper then looked at the “hot” ML trends for 2023, which examined the following:

- The IoT and the Manufacturing Processes;

- Natural Language Processing:

*The Differences between Syntactic and Semantic Analysis;

*The Use Cases such as: Sentiment Analysis, Systems that answer questions, Machine Translation.

- Applications to Cybersecurity:

  *Threat and Malware Detection;

  *Endpoint Security;

  *Protecting the Private/Hybrid/Public Cloud deployments.

In this whitepaper, we take a step further beyond 2023 and examine what ML even holds for then. The first topic that will be examined is that of Deepfakes.

The Use of Deepfakes

A Definition

Do you remember the Presidential Election back in 2016? Although they have been around for quite some time, this is the first time that Deepfakes had such an impact on our society. Essentially, it is the process of creating a fake image or fake video of an individual, and making it look like the real thing. Although it definitely has its entertainment value to it, it can also be used highly nefarious purposes as well by the Cyberattacker. But before we go any further, it is important to give a technical definition of it first:

"Deepfake is a form of artificial intelligence (AI) that can be used to create convincing hoax images, sounds, and videos. The term "deepfake" combines the deep learning concept with something fake. Deepfake compiles hoaxed images and sounds and stitches them together using machine learning algorithms. As a result, it creates people and events that do not exist or did not actually happen."

(SOURCE: 1).

So as you can see from the above definition, both AI and ML have a huge role in creating an effective Deepfake. Interestingly enough, these two can also be used to create a mock individual that does not even exist, but yet once again, make them look like the real thing. The definition also makes use of the term 'Deep Learning". Essentially, it makes use of the multiple layers that are typically found in an AI model. But the more hidden layers that there are, the more complex the AI model actually becomes. This is a direct correlation of the types of data feeds that are fed into the system. The more complex and unstructured the data sets are, the same will happen to the model.

The Algorithms That Are Used

In other words, simple data sets will lead to a simple AI model, but highly complex datasets will lead to a complicated one, which then in turn requires more layers to be produced. So, as it applies to Deepfakes, the more hidden layers that can be implemented into it will lead to that much of an authentic video or image, or even audio recording.

More specifically, the creation of a Deepfake makes use of what is known as a "Convolutional Neural Network", also known as a "CNN". This kind of algorithm consists typically of an input layer (in which the data feeds are plugged into), and an output layer (from which the result is computed, based upon the data that is fed and used into it). In between are the hidden layers, which processes the inputs. As stated before, the more complex that they are, more hidden layers will have to be created in return.

A subset of the CNN is known as the "General Adversarial Network", or "GAN" for short. Given the less complex nature of this algorithm, this is a very popular algorithm to use when creating a simple Deepfake video. In other words, it can be presented with a video of a real person, and from there, the output will be the almost replica image of it. The structure of the GAN is broken down as follows:

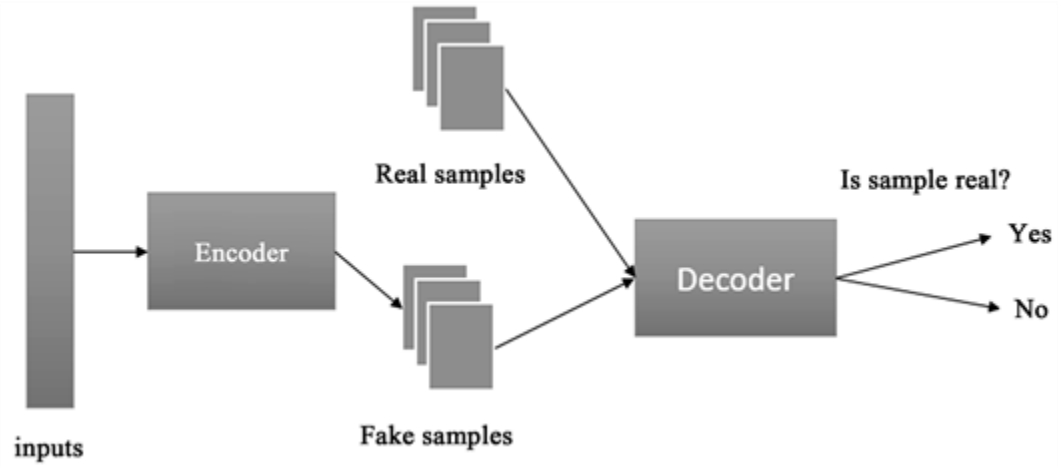
1) The Encoder:

   This is the component of the AI system that will ingest a huge amount of data, and attempt to learn from it. This is used to create the fake data, which is the output.

2) The Decoder:

   This component will learn then learn off of the fake data, making note of any strong similarities between that and the real data, in order to create the most realistic Deepfake video possible, which is the output. But keep in mind here that using a GAN system requires a gargantuan amount of data, and the datasets that are used must be the most optimized and cleansed as possible in order to produce very convincing fake. If not, the output will yield will some something far different, which will not look realistic at all.

An illustration of the GNA can be seen in the illustration below:



(SOURCE: 2).

One of the very first applications that has been created to produce a Deepfake video (or even just a picture is known as "FakeApp", which makes use of this Encoder-Decoder pairing technique. This can also be thought of as a supervised technique, because very close attention has to be given to the datasets that are used. In contrast, an unsupervised technique that can also be used to create a Deepfake is known as the "CycleGAN". For example, it can take the datasets that represents the eyes, and use that to create eyebrows and even forehead of the Deepfake. This is technically known as an

"unpaired technique" because there is absolutely no relationship to the input and the output. But you still need to have the datasets as optimized as possible so that a realistic looking correlation can be produced.

An illustration of a Deepfake image created from using the above techniques is illustrated below:

(SOURCE: 3).

How To Detect A Deepfake

Although the primary goal of Deepfakes is to create an image that looks like the real thing, AI and ML are just pieces of technology which are prone to some areas of imperfection as well. So, how can one tell if an image or a video is a Deepfake? Here are some of the clues to looks out for:

1) The face looks awkward:

   Look very closely. If you find that the nose is pointing slightly to the right and the lips to the left, for example, then this is a Deepfake.

2) Disjointed features:

   This is something to keep in mind for the Deepfake videos. If the face and the gestures of the person are not moving around smoothly, and there is some jerkiness present, then you have a Deepfake.

3) The coloring is not natural:

   Videos and pictures of real individuals will pretty much have natural lighting in them. But if you detect in any subtle differences, then they are Deepfakes. In fact, this is illustrated in the last diagram. It appears that the image to the right is the real one, and the one to the left is the Deepfake, because of the plain, dimmer lighting that is not present.

4) Noises that are present:

   In real video or picture, there should be no misalignment or blurriness present, which is referred to as "noise". If this is present, then you know what you have is a fake.

5) The audio does not match up:

   In this particular instance, if what the person is saying does not exactly coordinate with their lip movements, then the chances are high that you've got a Deepfake.

The Real Implications Of Deepfakes

As stated earlier, Deepfakes were originally created for entertainment purposes. But as the technology has further advanced using AI and ML, it is now being used for sinister purposes more than ever before, especially by the Cyberattacker. The greatest fear in this regard is during the Presidential campaigns. Deepfakes of the political candidates can be used to launch Phishing campaigns in which voters are lured to phony websites in order to donate money.

Deepfakes can also be used during times of national emergency, such as COVID-19. Phony videos of health officials can be created giving out false information, making a terrible condition into an even worst one. But probably where Deepfakes will be used most by the Cyberattacker is in Social Engineering attacks. For example, a fake phone call be placed to the administrative assistant of a member of the C-Suite, ordering him or her wire large sums of money to an offshore bank account. The voice created can be very authentic, and in fact, even mimic the real person.

Tiny Machine Learning

A Definition

Traditional AI and ML models need a lot of processing and computational power in order to produce the desired output, whatever that may be, such as a Deepfake as reviewed in detail in the last section. It's not because the algorithms are slow. By contrast, they are the component that requires the least of amount of resources. But what makes AI, and especially ML so hungry for resource allocation lies in the sheer fact that in order for them to learn, they need a huge amount of data, at least initially, to train on. This is where the drainage typically occurs.

But once this has been accomplished, the ML algorithms will still need a fair amount of resources if they are to run on a continual, daily basis. The use of Virtual Machines (VMs) that are created in the Cloud (such as that of the AWS or Microsoft Azure) can be designed to handle these gargantuan needs. But this can also be an expensive proposition for the business, as consumption costs will be very high, and this where most companies are charged at.

So, is there an alternative to this? Do we really need to have VMs running day and night to serve the ML models? Can we take just a subset of the data and perhaps the algorithm itself and use it on a smaller scale, so there is not so much resource consumption? Well, the answer to all of this is yes, there is a solution. It is called "tinyML".

A technical definition of it as follows:

"tinyML aka tiny ml is an abbreviation for tiny machine learning and means that machine learning algorithms are processed locally on embedded devices."

(SOURCE: 4).

The Microcontroller

This simply means that an ML algorithm, instead running on a full scale VM, can be run on a much a smaller piece of hardware, which is known as the microcontroller.

The next question that you may be asking, is “What is a Microcontroller”? Well, it is an individual integrated circuit board which is designed to serve one operation only. They are very small (hence the name they are given – “micro”), and in fact, quite cheap. Just one microcontroller can cost as low as .50 cents. An example of a Microcontroller is illustrated below:

(SOURCE: 5).

In fact, a microcontroller is like a Virtual Machine in many ways. For example, it consists of the following components:

1) A processor;
2) Memory (typically SRAM);
3) Input/Output ports;
4) A small sized CPU;
5) A small amount of hardware storage (just a few megabytes).

Interestingly enough, they don’t contain any networking components, and most of the microcontrollers depend upon an external source for power, such as that of a cell phone battery. In fact, most of them don’t even have an OS contained in them either. But one of the main advantages of the Microcontroller is that can run on very little power for long periods of time.

The fact that they are like their own computer means that ML algorithms can be run on it. Because they are also so small and portable gave rise to the name “tinyML”. Another succinct differentiation from tinyML versus the traditional means of processing ML algorithms via the VM is that the processing of the inputs and outputs can be done very close to the source that is requiring it. This means that the end user can get the results that they require in a shorter time period than the traditional ML/VM configurations. tinyML can also be deployed onto a smartphone, such as an Android or iOS device.

Thus, the end user really never has to connect to the Internet to get what they need. In a way, this is very comparable to what is known as "Edge Computing".

Key Components Of The tinyML

Other than the hardware components just described, the tinyML also has some of its own components as well, which need to be reviewed. These are as follows:

1) The tinyML software:

   These are the actual software packages that run the ML algorithms, and require very little power from the external source that is providing it.

2) The tinyML Framework:

   This is the platform from which the software developers can code and build out the tinyML software packages.

3) The tinyML Development Platform:

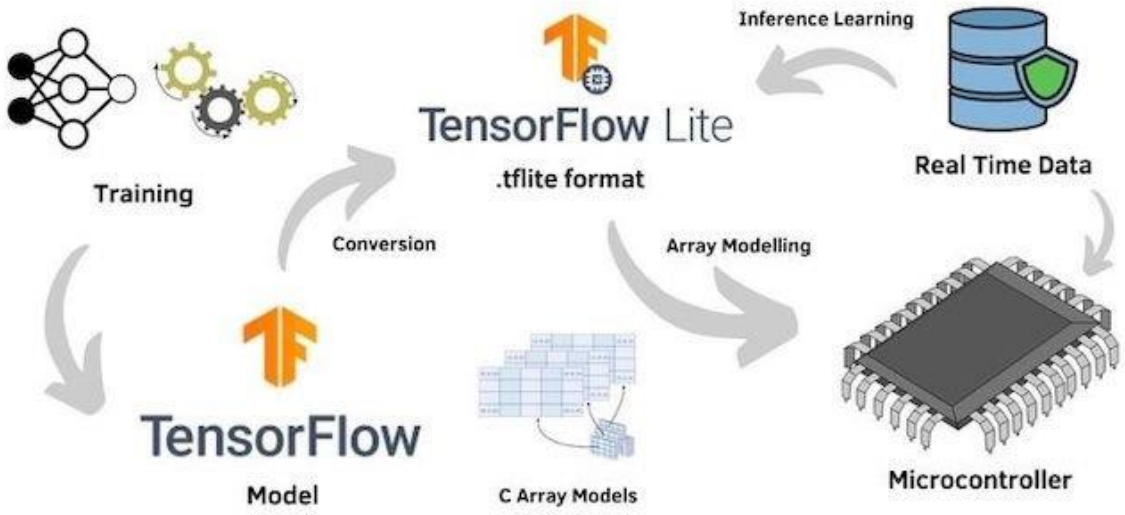
   This is where software developers can also develop customized applications to serve a specific ML purpose, such as predicting the outcome of a certain event, based upon the type of datasets that are being fed into it.

A Tool For tinyML

One of the most popular tools for developing tinyML software packages is known as the "TensorFlowLite For Microcontrollers". More detailed information about it can be seen here. The source code that is used to create the tinyML environment is Python, and it can be used to build out the following for any kind of application:

- Data acquisition (for feeding the datasets into);
- The preprocessing of the datasets to make sure that they are fully optimized and cleansed;
- The creation of the specific tinyML model (this will be primarily based upon the algorithms that are being used);
- Training (this is where the tinyML learn off of the inputs that have been ingested into it);
- Evaluation (has the desired output been yielded?);
- Optimization (this is where the refinements to the algorithms will be made, depending upon the output that has been obtained).

An illustration of the above workflow is depicted below:



(SOURCE: 6).

Typical Applications of tinyML

Some of these include the following:

1) Voice commands:

   The most typical of this are Siri and Cortana. In fact, they can even be considered as tinyML applications.

2) Images can be captured:

   A good example here is that of agriculture. Images of farm fields and crops are typically taken by drones, and sent back to the producer to their wireless device. An advantage of using tinyML here is that only can these pictures be analyzed to a very granular level, but based upon that, crop yields and harvesting patterns can be predicted.

3) The retail sector:

   A typical use case scenario is that of grocery stores. Future projections can be made using tinyML with wireless devices to predict at what time intervals commodities need to be purchased and stocked, so that shelves will always remain fully stocked.

4) Protection of farm animals:

   tinyML can also be used to protect the raw inputs of our food supply systems – namely the cattle and livestock. These animals be outfitted with a health monitor to relay their conditions back to the agricultural producer. From here, it can be determined if the animal is healthy, or is starting to get sick, based upon certain parameters that have been inputted into the tinyML application.

The Disadvantages Of tinyML

Despite all of the advantages that tinyML brings, there are some disadvantages to it also, which are as follows:

1) The memory is limited:

   Since Microcontrollers are very small in nature, their memory is very limited, thus limiting the sophistication and complexity that the tinyML software package run.

2) Training can be limited:

   Although tinyML software applications can be rather advanced in nature, training them at least initially can difficult because of the large amounts of datasets that have to be fed into them. This could require far more processing and computational power, than what the Microcontroller can provide.

3) Changes made:

   Since tinyML software applications are locally based, any changes made to it will have to be manually upgraded to other wireless devices that make use of them also. This is contrast to the traditional VM approach, where the updates can be made in one location, and from there, pushed own to the devices simultaneously.

Conclusions

Our next whitepaper will examine another future trend of ML – The Digital Twins, and Robotic Process Automation.

Sources

1) https://www.fortinet.com/resources/cyberglossary/deepfake
2) https://www.scirp.org/journal/paperinformation.aspx?paperid=109149
3) https://vitalflux.com/how-to-create-detect-deepfakes-deep-learning/
4) https://www.imagimob.com/blog/what-is-tinyml
5) https://www.imagimob.com/blog/what-is-tinyml
6) https://www.allaboutcircuits.com/technical-articles/what-is-tinyml/
7) https://www.mygreatlearning.com/blog/all-you-need-to-know-about-deepfake-ai/