Exploiting HDMI and USB Ports for GPU Side-Channel Insights

Sayed Erfan Arefin *Department of Computer Science*
Texas Tech University, Lubbock, Texas
saarefin@ttu.edu
Abdul Serwadda *Department of Computer Science*
Texas Tech University, Lubbock, Texas
abdul.serwadda@ttu.edu

***Abstract*—Modern computers rely on USB and HDMI ports for connecting external peripherals and display devices. Despite their built-in security measures, these ports remain susceptible to passive power-based side-channel attacks. This paper presents a new class of attacks that exploit power consumption patterns at these ports to infer GPU activities. We develop a custom device that plugs into these ports and demonstrate that its high-resolution power measurements can drive successful inferences about GPU processes, such as neural network computations and video rendering. The ubiquitous presence of USB and HDMI ports allows for discreet placement of the device, and its non-interference with data channels ensures that no security alerts are triggered. Our findings underscore the need to reevaluate and strengthen the current generation of HDMI and USB port security defenses.**

***Index Terms*—Article submission, IEEE, IEEEtran, journal, LaTeX, paper, template, typesetting.**

I. Introduction

Today's computers are equipped with numerous Universal Serial Bus (USB) and High-Definition Multimedia Interface (HDMI) ports to support the wide range of external peripherals and visualization needs of modern users. These ports provide a physical channel to the underlying hardware and software, making them integral to the computer's threat surface. As such, they are built with security features designed to prevent exploitation. For instance, USB ports disable the autorun feature to prevent malware on a USB-connected device from executing automatically. Operating systems also notify users whenever a USB-connected device attempts to access the file system, allowing users to block unexpected actions [1]. HDMI ports, primarily output devices, do not support writing to the computer, further reducing their susceptibility to certain types of attacks. These and other security measures help safeguard these physical ports from becoming vectors for attacks [2] [3].

In this paper we argue that current port security defenses are inadequate in the wake of passive attacks that leverage these ports as a vehicle to drive power measurements designed to infer key underlying processes on the computer. Using the example of processes running on the GPU, we demonstrate the viability of these attacks through the development of a custom device that plugs into USB and HDMI ports, facilitating inference attacks on GPU activities.

Our approach is stealthy and does not interfere with the processes that typically trigger user alerts. For the USB port, our attack is designed to avoid manipulating the data channel, thereby bypassing security notifications. For the HDMI port, our device mimics a conventional display device, ensuring it does not trigger any errors or alerts. The ubiquitous presence of USB and HDMI ports on contemporary computers – located at the back, front, and sides – facilitates the discreet deployment of our attack devices. A small, inconspicuous device plugged into any of these ports can remain unnoticed, providing an effective means of conducting attacks without installing software on the victim's computer or attaching hardware to internal components.

The overarching idea behind our attack is as follows: despite modern computers being equipped with over-provisioned power supplies and sophisticated stabilization mechanisms to prevent power instability, significant power draw increases by a GPU during intensive processes can induce detectable fluctuations in power seen by peripheral devices attached to USB or HDMI ports. These fluctuations, though transient and eventually stabilized, contain discernible information sufficient to infer GPU operations. Using GPU loads in the form of matrix multiplications, video scene rendering and neural network execution, we demonstrate that this line of attack poses a potent side-channel that can leak sensitive information about processes running on the GPU.

A. Paper Contributions

Our research makes the following Contributions:

(1) GPU power side-channel inference via HDMI and USB ports: Our research introduces a novel family of power side-channel attacks, which utilize power consumption patterns observed at USB and HDMI ports to infer computational processes occurring on a GPU. This method is distinct from traditional attacks that measure power directly from the GPU, as it exploits peripheral connections that either interface with or share a power source with the GPU. Requiring only the insertion of a device into a peripheral port, this stealthy approach operates at the user's privilege level without triggering any alerts. This technique not only broadens the scope of implementable side-channel attacks but also suggests the potential for using peripheral power measurements defensively, such as detecting unusual power draw signatures that indicate malicious activities.

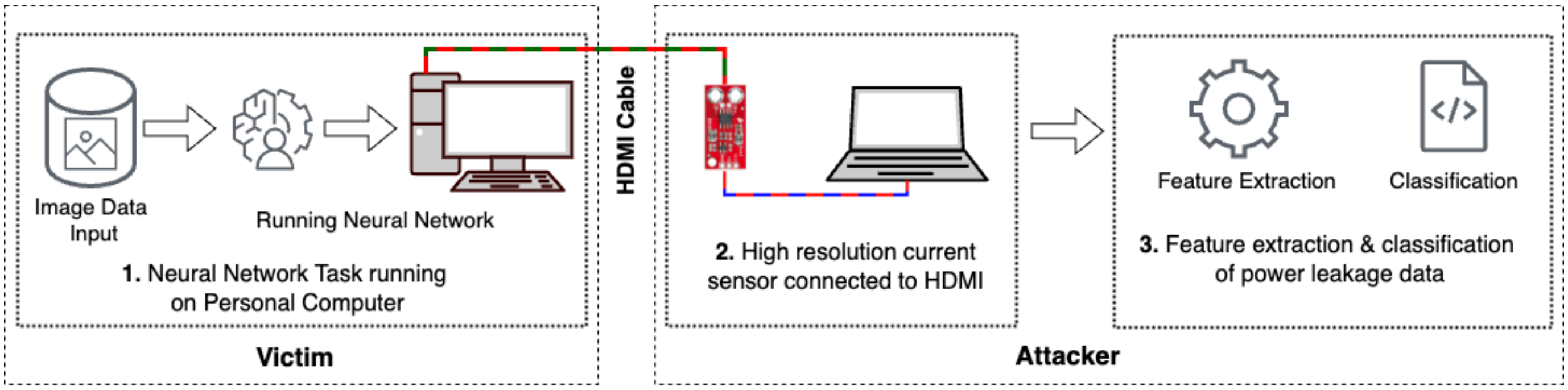

Fig. 1: Illustration of the attack process. The example used here is that of a neural network running on the victim/targeted computer

(2) Design of power measurement prototype: We present the design of the power measurement tool central to our attack strategy. At the core of this tool is a SparkFun ACS723 Current Sensor, which utilizes a Hall effect sensor to produce a voltage output correlating to the current through its measurement pins. This sensor is integrated with an Arduino board equipped with a 32-bit ARM Cortex M0+ processor, enabling data collection at a rate of up to 350,000 samples per second. This high-resolution data is crucial for driving the machine learning classifiers that discern and classify GPU activities based on power fluctuations.

(3) Evaluation of attack performance: Our evaluation of the attack's performance begins with an intuitive analysis of visual patterns in time-series data extracted from GPU computations. This preliminary analysis provides initial indications for the feasibility of the attack, leading to the development of machine learning models capable of executing two distinct attack scenarios with high accuracy. In the first scenario, our side-channel approach successfully identifies the specific family of deep neural networks running on the GPU. In the second, it accurately infers which video clips are being rendered by the GPU. These examples demonstrate the broader applicability and effectiveness of our side-channel techniques in targeting GPU-based processes.

B. Threat Model

Our threat model assumes an attacker who has physical access to a target computer but lacks login credentials. Such individuals might include office cleaners, maintenance staff, or rogue employees who can access the machine without raising suspicion. The attacker can discretely insert a small power measurement device into an inconspicuous USB or HDMI port on the computer, preferably one that is not easily visible (e.g., the ports at the back). This device is designed to operate stealthily, drawing power data without causing any disruptions or triggering the alerts typically associated with peripheral connections like USB storage devices. The device does not interfere with normal computer operations, such as display functions when connected to an HDMI port.

The attacker's goals may vary from stealing power measurement data linked to high-value algorithms, such as neural network architectures or cryptographic operations, to general surveillance of the user's GPU-dependent activities, which might include video rendering or website browsing. To facilitate the machine learning aspect of the attacks, the attacker would need to acquire training data that reflects the specific computational processes targeted.

In our laboratory setup, we utilized a wired connection for data transfer, sufficient for demonstrating the attack's foundational concepts. However, in a practical application, this setup could be easily modified to include wireless data transfer, such as using Bluetooth or Wifi to transmit data to a device in a nearby location. Figure 1 captures the attack process, taking the example where the attacker seeks to infer information about a neural network architecture running on the victim's device.

C. HDMI/USB Power Side-Channel for Defense

While the primary focus of our paper is on exploiting the described side-channel for attacks, it is important to note its potential defensive applications. Specifically, this technique could be useful for anomaly detection or malware identification through deviations in power consumption patterns. For example, a computer infected with malware that engages in resource-intensive activities unknown to the user — such as crypto-mining, data exfiltration, or unauthorized network attacks — would exhibit abnormal power usage. Employing a USB-type or HDMI-type power measurement device can provide a significant advantage in such scenarios. It operates independently of the computer's software, making it less susceptible to tampering by malware that might disable or manipulate software-based monitoring tools. Furthermore, its simple installation and ease of use make it accessible to users without specialized technical skills, offering a non-invasive, reliable method for monitoring system integrity.

II. Related Works

Our research has two clusters of related work: power side-channel attacks on computing devices, and attacks that exploit vulnerabilities of I/O devices. In this section, we discuss these two families of research and describe how we differ from them.

A. Power side-channel attacks:

The wide array of past studies on power side-channel attacks measured power in several different ways depending on the

threat scenario and targets of the attack. We categorize these related works based on the ways in which power was measured before discussing how our work differs from this body of past research.

Using Nvidia-smi interface: Nvidia-smi is a widely used interface to measure nvidia GPU resources, including power consumption [4]. The interface utilizes the nvml library [5] provided by Nvidia GPUs. The research conducted by Jha et al. [6] investigates a two-stage attack methodology designed to infer the architecture of deep neural networks (DNN). The study utilizes nvidia-smi to analyze performance on P100 and P4000 GPUs, demonstrating the attack's effectiveness. The authors proposed a secure MobileNet-V1 as a mitigation to such vulnerability. In their work Sabbagh et al. [7] presents an overdrive fault attack on modern GPUs, exploiting nvidia-smi to introduce random faults during kernel execution. The authors effectively recovered AES keys in their experiments. The study by Sayed et al. [8] explores power consumption patterns using nvidia-smi to determine if this data could reveal key details about DNN architectures running on GPUs. They conducted model inference attack on several nvidia GPUs including some cloud GPUs.

Using Intel's Running Average Power Limit (RAPL) interface: RAPL is the primary interface for power measurement on Intel CPUs. Moritz et al. [9] introduced a software-based power side-channel attack that leverages unprivileged access to this interface. The authors successfully extracted cryptographic keys and monitored application control flows without needing physical access or specialized hardware. Demonstrations included breaching Intel SGX to extract AES-NI keys, and recovering RSA keys via privileged attacks. Xiang et al., [10] conducted a research where they introduced a novel approach to model extraction in deep learning using the Intel Running Average Power Limit (RAPL) to exploit power leakage in ReLU activation functions via a software interface. Their attack used a very small number of queries to extract deep learning models by leveraging software-based power side-channel information. This technique was implemented on the oneDNN framework. They extracted a 5-layer MLP and Lenet-5 CNN. The study conducted by Segev et al., [11] showed how performance counters can be used to infer information about neural networks. They used the Intel Power Gadget to collect detailed performance traces, including power consumption and CPU utilization. The software used the Intel RAPL technology in its core.

Exploiting Dynamic Voltage and Frequency Scaling (DVFS): DVFS is a power management mechanism used in Intel, AMD, and ARM CPUs, that dynamically adjusts CPU frequency and voltage to reduce overall system power consumption and enhance performance per Watt. While DVFS helps in managing energy efficiency, it also introduces a potential vulnerability that can be exploited through power side-channel attacks. For example, by observing the power consumption variations that occur as a result of DVFS adjustments during the computation process. Chen et al., [12] successfully extracted AES keys by leveraging the DVFS mechanism, while Yingchen et al., [13] studied how power side-channel attacks on modern Intel and AMD x86 CPUs can be used via Dynamic Voltage and Frequency Scaling (DVFS), without requiring access to a power measurement interface directly. Debopriya et al., [14] used the DVFS side-channel attack to perform website fingerprinting for browsers like Google Chrome and Tor across various CPU architectures, while Sehatbakhsh [15] manipulated the power management units to enable side-channel attacks through electromagnetic emanations from the voltage regulator module (VRM). They utilized this technique to execute keystroke logging.

Direct power measurement using an oscilloscope: Hasindu et al., [16] outlines a power analysis attack, where they focus on extracting cryptographic keys from devices by analyzing power consumption data during cryptographic operations. The researchers demonstrate direct power measurement using an oscilloscope. They utilize power analysis techniques to break the Speck algorithm for embedded systems. Luo et al., [17] study the power side-channel on Graphics Processing Units (GPUs) to extract a secret key from a block cipher operating on a GPU. In order to capture power they inserted a resistor in series with the PCI-Express power supply and measured the power physically. They targeted a CUDA AES implementation on an Nvidia Tesla GPU.

Phone charging stations: Yang et al., [18] highlighted the privacy risks posed by public USB smartphone charging stations whose public placement makes them vulnerable to malicious entities that rigg them with power measurement functionality. Using one such rigged hub, the researchers demonstrated a side-channel attack that can identify web pages loaded on a smartphone during the phone charging process. They conducted their research with different browser settings, network types, and battery levels, revealing successful attacks under all these settings. Sraddhanjali et al., [19] extended this line of attack and demonstrated that USB charging hubs can analyze power consumption data to infer YouTube music videos that a user is watching on their smartphone, while Alexander et al., [20] focused on the wireless charging interfaces on modern smartphones. They showed that the wireless charging interface is also susceptible to power-side-channel attacks, specifically demonstrating a website fingerprinting attack on iOS and Android devices.

Other power measurement approaches: Zhao et al., [21] showcased a new security vulnerability introduced by integrating Field Programmable Gate Arrays (FPGAs) into cloud data centers and System-on-Chips (SoCs). Their work reveals that FPGAs can facilitate software-based power side-channel attacks remotely without the need for physical access to the target system. The researchers demonstrate using ring oscillators on FPGAs to monitor power consumption of both FPGA modules and CPU components within the same SoC, successfully executing power analysis attacks on RSA cryptomodules and bypassing timing-channel protections for CPUs. Sayed et al. [22] demonstrated a power side-channel attack on the GPU to infer deep neural network architectures using a software-based approach. Their attack reads power consumption data from the on-board sensors of the GPU.

How we differ from these works: The fundamental difference between these studies and our research is that we introduce a novel side-channel – specifically, power leakage

through peripheral devices such as the HDMI and USB ports. While previous works have relied on either invasive physical measurements directly from the power supply or sophisticated software approaches that require specific system access or vulnerabilities, our approach works by methodically triggering a power draw from the HDMI/USB ports that then enables inferences on the underlying processes running on the GPU. This is achieved without altering the device or needing special privileges.

Notably, our method capitalizes on the inherent power variations that pass through these peripheral ports as a byproduct of GPU processing loads, making it a unique point of observation that remains largely under-explored in current literature. Our technique, therefore, opens new avenues for both understanding and mitigating security risks in modern computing environments where external peripherals are common.

On the surface, the charging station attacks (e.g., [18]–[20] might seem to have some similarity to our work given that they also use a USB port. However, that attack is fundamentally different from our attack since the target of the attack is the phone plugged into the USB port, and the power being drawn by the phone is directly measured by a circuit placed between the phone and the USB port. Inferences about activities being undertaken on the phone are then done based on these direct power measurements. In our attacks on the other hand, the USB (and HDMI) ports are simply a subset of the many peripherals that share a power source with the GPU, indirectly picking up GPU power patterns through fluctuations seen in its own power as supplied by the source. These indirect power patterns are then used to infer information about computations that occur on the GPU.

B. *I/O port related vulnerabilities:*

The body of past research on I/O security vulnerabilities related to our research is mainly focused on the USB port, given its usage for a wide range of device connectivity applications. We only briefly review these works, given that they do not focus on the power side-channel which drives our work.

Ramadhanty et al. [23] investigate the vulnerability of Windows 10 to USB-based keylogger attacks. By embedding a PowerShell script in an Arduino Micro, they demonstrated a successful keyboard injection attack. Pham et al. [24] examine the information security risks introduced by USB storage devices due to their insecure design and open standards. Their work reviews various USB-based software attacks on host computers and USB devices, exploiting vulnerabilities in USB protocols, embedded security software, drivers, and Windows Autoplay features.

Nohl et al. [25] investigate an attack in which USB firmware is reprogrammed to bypass traditional security measures and execute unauthorized commands, while Jeong et al. [26] analyzes the vulnerabilities of popular secure USB flash drives. They found that passwords can be exposed during communication between the USB drive and the PC. Furthermore, the study reveals that some secure USB flash drives are susceptible to firmware attacks, allowing unauthorized users to bypass security measures.

Brian et al. [27] show how the USB Switchblade could be used to steal sensitive information such as passwords, network configurations, and system details, making it a powerful tool for attackers to gain unauthorized access and steal data from a target machine. Kaspersky Lab [28] discusses scenarios in which a compromised USB storage device can be used to secretly extract sensitive data from a victim machine. This process occurs without the awareness of the host or the USB device owner.

Tariq et al. [29] review cybersecurity challenges in IoT, focusing on vulnerabilities in USB-connected devices. They discuss the risks of malware spreading through USB ports and emphasize the need for securing these interfaces. The authors highlight the importance of developing robust security measures to mitigate these threats, suggesting the use of better authentication protocols and noting that traditional security measures are inadequate for modern IoT environments.

How we differ from these works: While previous works share our objective of highlighting the security threats posed by USB devices, our approach is distinct in that it focuses on a passive power measurement attack. Unlike these prior attacks, our method does not require any special privileges or engage in any communication with the underlying operating system or applications. Instead, it passively measures power consumption patterns, making it a stealthier and less intrusive method compared to attacks that actively probe the target system.

III. Basics of Computer Power Supply Mechanisms

To provide context to our attack design, this section covers some basic information about the power supply mechanism is computers. We first discuss the power supply unit (PSU) and how power is supplied to different peripherals of the computer. We finally discuss some input and output ports (I/O ports) that can potentially power external devices.

A. *Computer Power Supply Unit (PSU)*

Often the power supply used in a home-usage, gaming computer or a workstation is an ATX power supply. A sample picture of an ATX power supply is shown in Figure 2b [30]. Various power cables are used to connect all the components to the power supply. Each type of connector has specific current and voltage ratings. A very widely used connector is the 24-pin ATX power connector, which connects to the motherboard. It can deliver multiple voltages, such as +3.3V, +5V, and +12V. The 24-pin connector can supply significant current, up to 24A on the +5V rail and 24A on the 3.3V rail. However, this current supply depends on the power supply's capacity. The SATA power connector is used to power devices with a SATA interface such as hard disk drives (HDDs), solid state drives (SSDs), or optical drives (CD/DVD drives). This connector supplies three voltages: 3.3V, 5V, and 12V. The CPU power connectors, typically known as the 8-pin connector, connect to the motherboard to supply power to the main processor. This connector can provide +12V to the CPU. Today, most

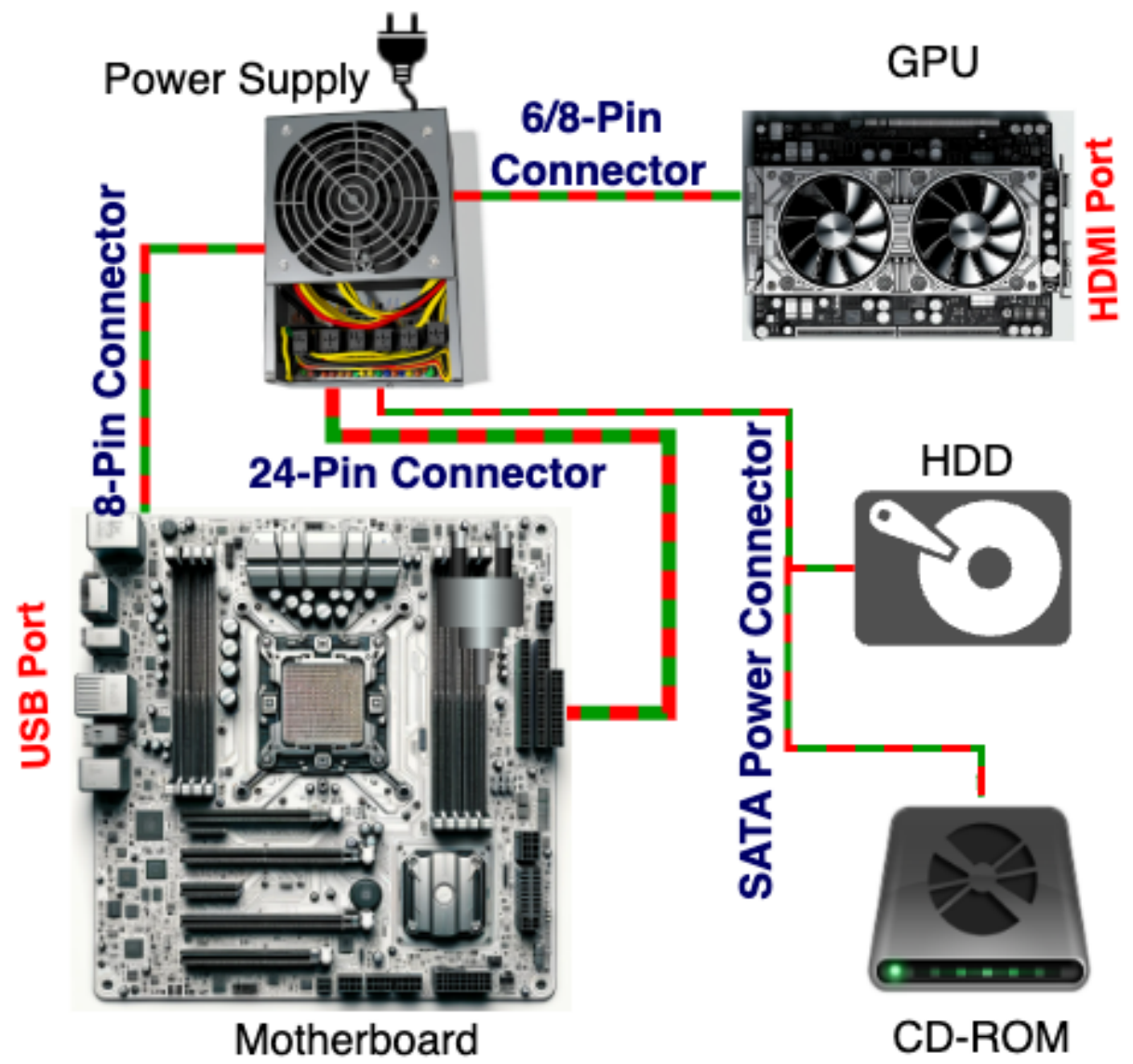

(a) How power is supplied inside the computer components.

(b) A sample ATX Power Supply

Fig. 2: Desktop Computer Power supply.

computers are equipped with high-performance graphics cards. These are used for advanced graphics processing and machine learning tasks. The 6-pin or 8-pin PCIe connector can deliver high power to GPUs [31]. An illustration of the power supply connectors and the corresponding computer components is presented in Figure 2.

B. Computer I/O ports

There are several input and output (I / O) ports in the computer that can be used to connect different peripherals. These peripherals can be a wide variety of devices. In case of USB devices, keyboard, mouse, external hard disk drive, etc. are a few notable peripherals. These devices require power to operate, which is consumed over the USB port. The current flow from a regular USB port can vary depending on the specific USB standard and the device providing the power. For USB 2.0, the standard current output is typically 500 mA (milliamperes) at 5 volts, providing a maximum power of 2.5 watts. For USB 3.0 and later versions, the current output can be higher, often up to 900 mA or more, again at 5 volts. In our experiments, we used the USB 2.0 ports.

We also observed the current draw made by the HDMI port. The HDMI port is in several ways a more complicated port than the USB port. It does not supply active power to the display devices, as those require external power. However, it has a power saving option, where the device goes to sleep if there is no signal in the HDMI port. This is called hot plug detection (HPD). There is a small circuit that periodically checks for the signal such that, when there is a signal, the display device can come up from its sleep status. In order to run this small circuit, a small amount of power is supplied over the HDMI port. Pin 18 of the port supplies 5v and at most 500 milliampere, to run this circuit. In our experiments described in Section V we monitor the 18th pin of the HDMI port and the power pin of the USB port. Details of the sensor and circuit used to make measurements on this pin are discussed in Section IV-A.

IV. Attack Design

In this section of the paper, we discuss our hardware setup to measure current and possible power leakage from the I/O ports. The following sections describe the circuits used and the hardware setup in Section IV-A. High resolution data collection required certain setup for the analog-to-digital converter (ADC), which is described in Section IV-B. In addition, the computer used to run our experiments is described in Section IV-C.

A. Hardware setup

At the heart of our hardware lies a current sensor capable of recording the current passing through a series circuit. The hardware setup incorporated a SparkFun ACS723 Current Sensor [32]. This module uses a Hall effect sensor to generate a voltage output that corresponds to the current passing through the measurement pins on the board. Since it uses a hall effect sensor, it ensures electrical isolation between the monitored circuit and the measuring circuit. The module can measure a current from 10mA to 5A. It has an onboard preamp to adjust the resolution of the readings. We had increased this to an optimum point to have a better resolution of the reading. The analog output of the module is set to 20kHz in order to reduce noise as the gains are set to high. The sensor is connected to an Arduino to read the analog signal and produce a digital reading to be used later with machine learning classifiers. In our experiments, we used an Arduino MKR Zero board. This board is equipped with an Atmel SAMD21, which uses a 32-bit ARM Cortex M0+ processor [33].

The analog-to-digital converter (ADC) can achieve sampling rates as high as 350,000 samples per second. The sensor has 3 pins. These are the +5V, ground, and analog output pins. The analog output pin of the sensor is connected to the analog input pin (A2) of the arduino. The ground and 5V are connected to the corresponding power terminal of the circuit. The sensor board is equipped with an opamp and other operational components. The final assembly of the whole circuit can be seen in Figure 3. In order to measure the current, we need to place the current sensor in series. But, since we want to measure the current draw of the I/O ports, we cannot put it in series directly, but rather have a load in the series.

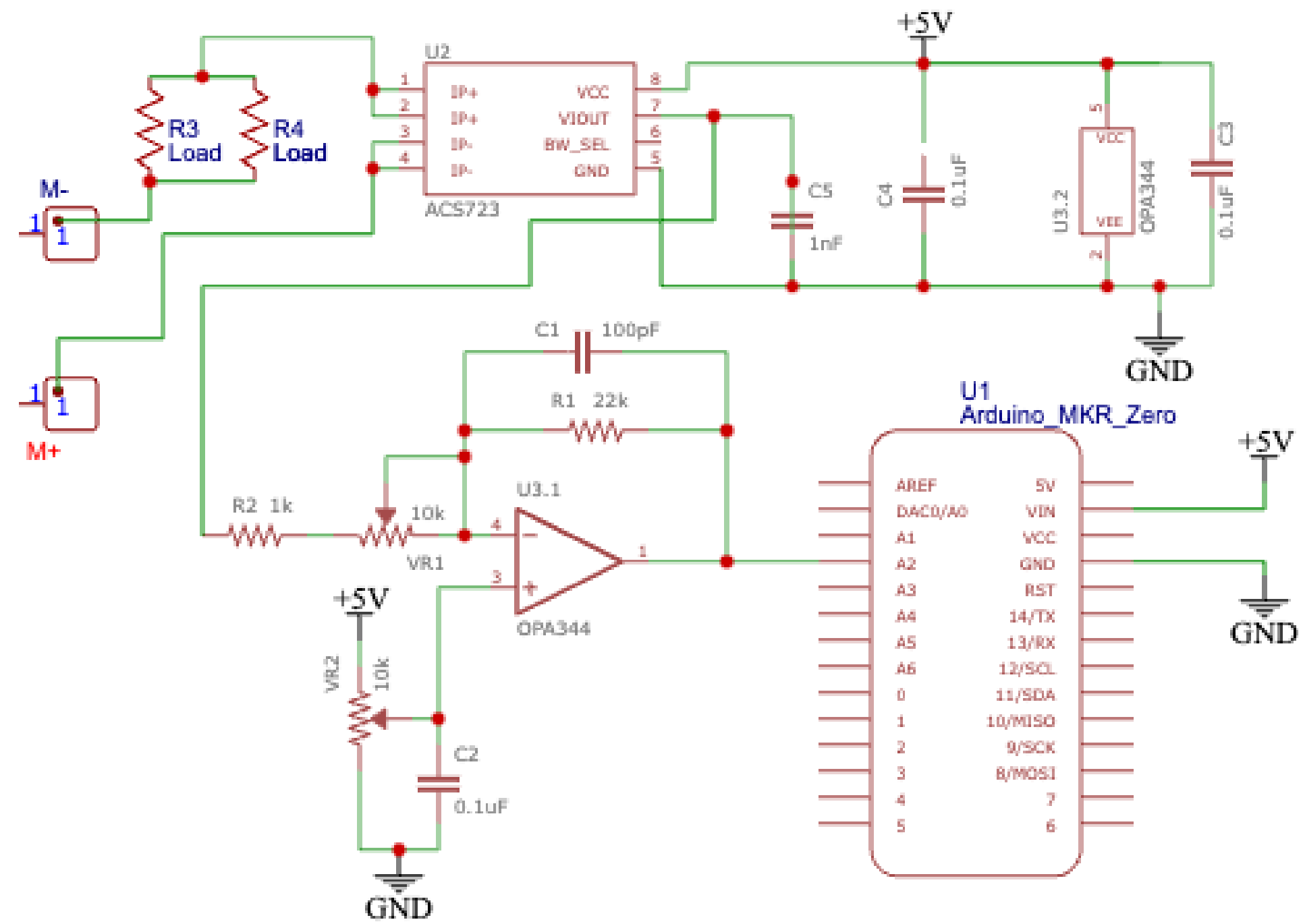

Fig. 3: Circuit of the attack device.

Without the load, the short-circuit protection of the I/O boards may trigger.

Power dissipation in resistors generates heat. Distributing the power dissipation across multiple resistors can help to better manage heat, preventing any single resistor from overheating. In our experiment, we had two resistors in parallel. Each resistor in our experiment was a 20Ω film resistor. These resistors are marked as load R3 and R4 in the circuit presented in Figure 3. The final assembly of our attack device is presented in figure 4 which labels the components of the device. The placement of the current sensor on top of the Arduino can be observed. The figure also labels two cables. One will be connected to the victim computer to measure power. The other cable is connected to the attacker computer.

In Section I-B, we discussed the possibility of enhancing the attack device by integrating a Bluetooth or WiFi module, thereby extending the operational range of the attacker. In this advanced scenario, the cumbersome cable to the attacker device becomes obsolete, replaced by seamless wireless communication. The attack device, in this extended setup, would draw power from an independent source, such as a 5V battery.

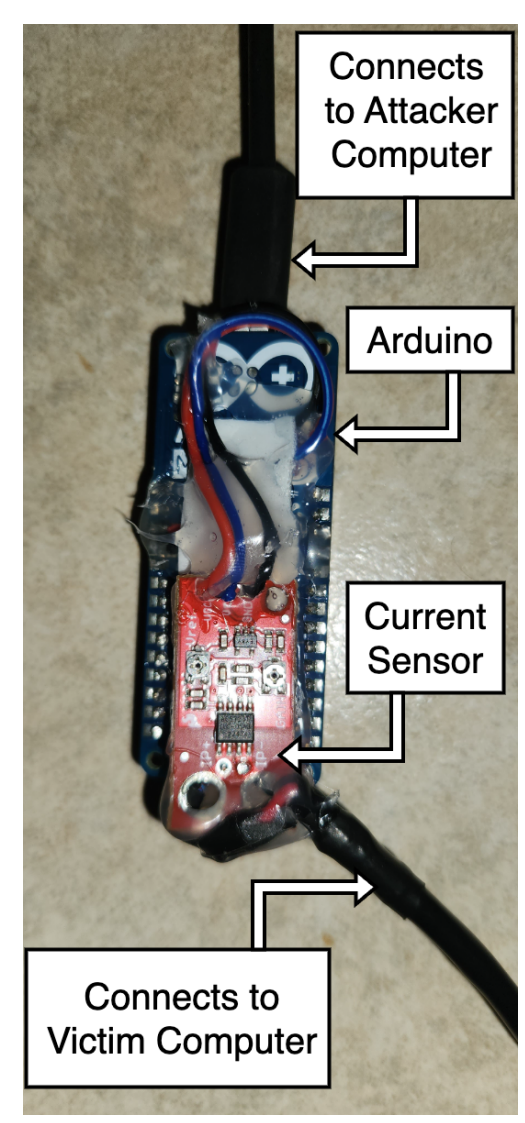

Fig. 4: The attack device.

B. Arduino Implementation Details

In our Arduino software setup, we perform high-speed data acquisition from an analog current sensor. Our objective is to capture the analog data and transmit them to a computer, where they are saved as text files (.txt) using a Python script. To facilitate rapid data transfer, we configure the Arduino's serial communication to operate at a baud rate of 2,000,000 bits per second. We adjust the Analog-to-Digital Converter (ADC) to optimize data quality and sampling rate. The ADC resolution is set to 12 bits to enhance the fidelity of the digital conversion.

As mentioned earlier, the ADC samples from the analog pin A2, which connects to the current sensor's output. Our code uses continuous sampling with interrupts, allowing the ADC to function efficiently with minimal processor overhead. We configure the ADC's control register B (CTRLB) to set the clock prescaler to a 40 kHz operating frequency. This configuration achieves a sampling rate of 40,000 samples per second, which is well above the required 20,000 samples per second for our current sensor application. An ADC Interrupt Service Routine (ISR) is programmed to handle the data conversion results. It reads the ADC output, transmits it via USB, and resets the ADC for the next conversion.

C. Experiment Machine

The experiments were carried out on a desktop computer that had the following configuration. The graphics processing unit (GPU) used in the computer is an Nvidia RTX 2060 Super with 8GB graphics memory. This uses the Nvidia Turing Architecture. The other components of the computer include a Central Processing Unit (CPU) Intel Core i7 9600K Processor, 16 GB RAM, 4 TB hard drive (HDD). The operating system (OS) used is Ubuntu 22.04 LTS. The tensorflow version used in the experiment is a tensorflow-gpu 2.12. The computability of the tensorflow setup is 7.5. The power supply used in the computer is a 450 watt power supply. The GPU uses a 6-pin power connector. GPU requires a 175 Watt power and CPU requires a 95 Watt power. The other components draw a variable power based on the need.

V. Attack Experiments

In this section of the study we discuss different parts of the attack experiments. At first we conduct several preliminary experiments and observe the power leakage phenomenon (Section V-A). Afterwards, we discuss how we conducted data collection and experiments to infer the Convolutional Neural Networks model architecture with several levels of experiment configuration (Section V-B1). We also performed experiments in inferring video from several video-rendering tasks (Section V-B2). In the later part of this section, we discuss the datasets used in our CNN model architecture inference experiment (V-C1) and video inference experiment (V-C2).

A. Preliminary Experiments

Before undertaking a detailed design of the attack, we first conducted a preliminary experiment to discern if GPU operations produce measurable current draw fluctuations at our attack device. In the first of these preliminary attacks, we used the Glmark 2 software [34] to render videos on the GPU. This software spawns a window and renders scenes that make use of OpenGL (ES) 2.0 and is commonly used to benchmark GPU performance. Before collecting any data, we ran the command to see GPU activity using the Nvtop tool [35] and observed some GPU utilization. Figure 6 captures the findings from this experiment for one of the scenes. The figure reveals a slight hump in between the 1 and 2-second marks when the rendering happens. For both the USB and HDMI ports, our device is hence able to pick up the power fluctuations triggered by the video rendering on the GPU. Similar patterns (not shown here) were observed for various other scenes, providing us with the first signs that our line of attack might be feasible.

Our second preliminary experiment focused on matrix multiplication, a fundamental computation underlying many GPU operations such as image processing and neural network computations. We aimed to observe power leakage from the I/O ports during different types of matrix multiplication.

To formulate this experiment, we used matrices of size 1920 x 1080, matching the resolution of a typical full HD display. This setup mimics a regular use case where a matrix of this size is executed on the GPU. The goal was to multiply two matrices on the GPU, ensuring execution via the CuPy library [36], which utilizes the GPU for matrix manipulation operations.

We created four matrices, A, B, C, and D, with random values between 0 and 255 to replicate the color values of each pixel. We then performed two matrix multiplication tasks: A multiplied by B, and C multiplied by D. Each multiplication task was repeated 25 times to observe the power leakage phenomenon. Following the multiplications, we extracted features, specifically the mean and MFCC 2 (Mel-frequency cepstrum coefficient) of the time series data, and plotted these features.

This experiment was conducted for both HDMI and USB I/O ports. The feature plots are presented in Figure 6. The plots show that our USB and HDMI measurements depict distinct clusters of power measurement data for the two different matrix multiplication tasks conducted on the GPU. Given these positive results from the two toy experiments, we proceeded to design and execute the fully-fledged attack experiment that we discuss next.

Base Name	Variants
ConvNeXt	ConvNeXtBase, ConvNeXtLarge, ConvNeXtSmall
VGG Net	VGG16, VGG19
ResNet	ResNet50, ResNet152V2, ResNetRS420
InceptionNet	InceptionV3, InceptionResNetV2
MobileNet	MobileNet, MobileNetV2
DenseNet	DenseNet121, DenseNet169, DenseNet201
NASNet	NASNetMobile, NASNetLarge
RegNet	RegNetX002, RegNetX160, RegNetY320
EfficientNet	EfficientNetB0, EfficientNetB5, EfficientNetV2L
XceptionNet	(No variants)

TABLE I: Base CNNs and their Variants used in our Experiments

B. Attack Configurations and Data collection

In this part of the research, we focus on the detailed experiments of our research. We first run experiments to infer Convolutional Neural Networks and later infer video rendering.

1) Inference of CNN Model Architecture running on the GPU: Here we focus on the inference of the architecture of the convolutional neural network (CNN) using the power monitoring from the I/O ports. The CNNs are now widely used in many applications for a wide range of applications such as Image classification, Object detection, Video analysis, etc. The victim runs a CNN to classify the image, and the attacker will monitor the I/O port and try to infer the architecture of the CNN. We conducted this experiment for both HDMI and USB I/O ports. In this meticulously designed experiment, we postulate that the attacker has ingeniously adapted well-established CNN models and has proficiently trained machine learning models using the I/O port power data. Upon acquiring I/O port power data from the victim's machine, the attacker is equipped to precisely classify the specific CNN that was operational on the victim's system. Our investigation focuses primarily on the CNN architectures listed in Table I, all of which benefit from the availability of pre-trained models in Tensorflow [37].

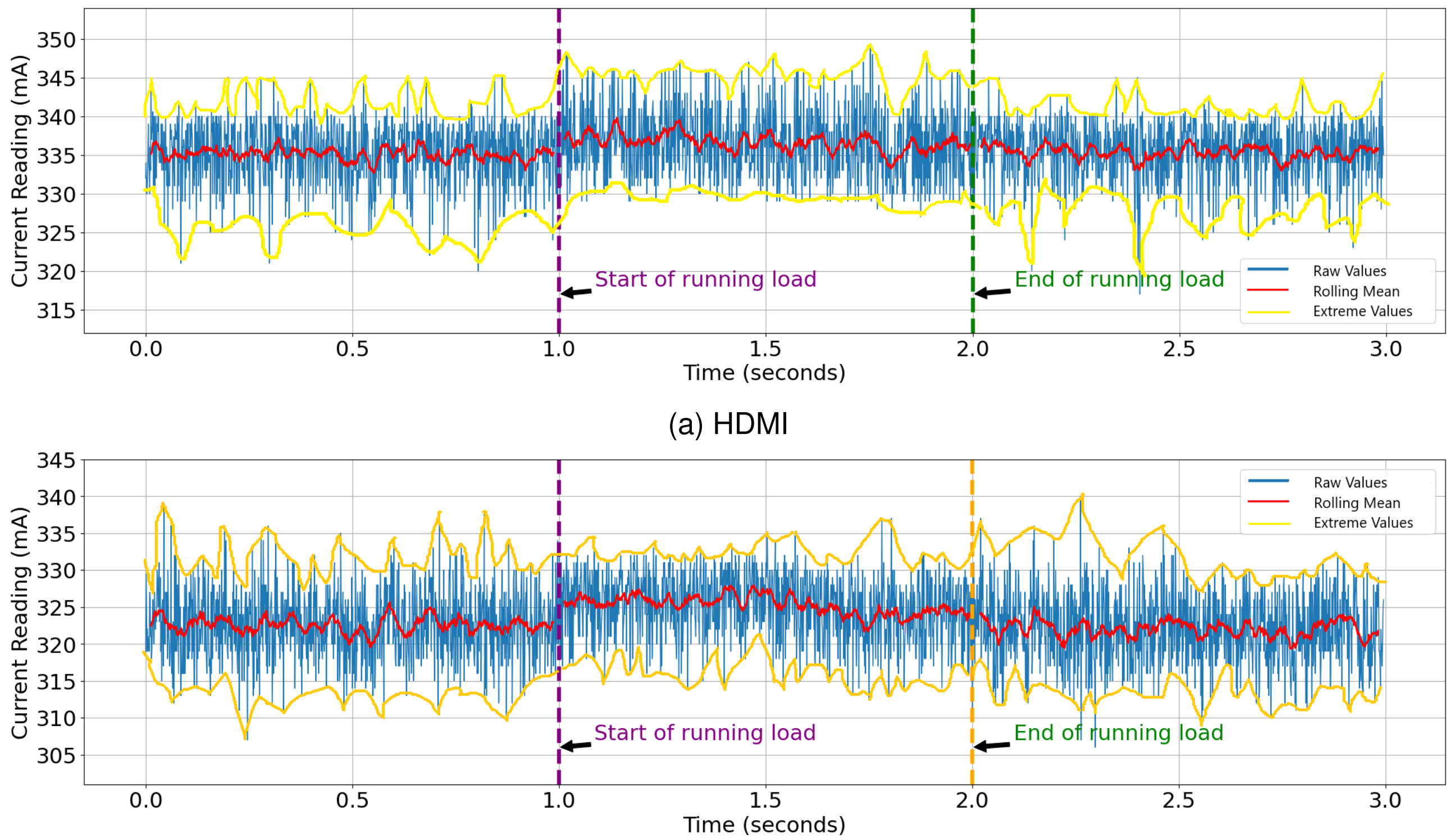

(a) HDMI

(b) USB

Fig. 5: Raw reading for Scene rendering with Glmark software.

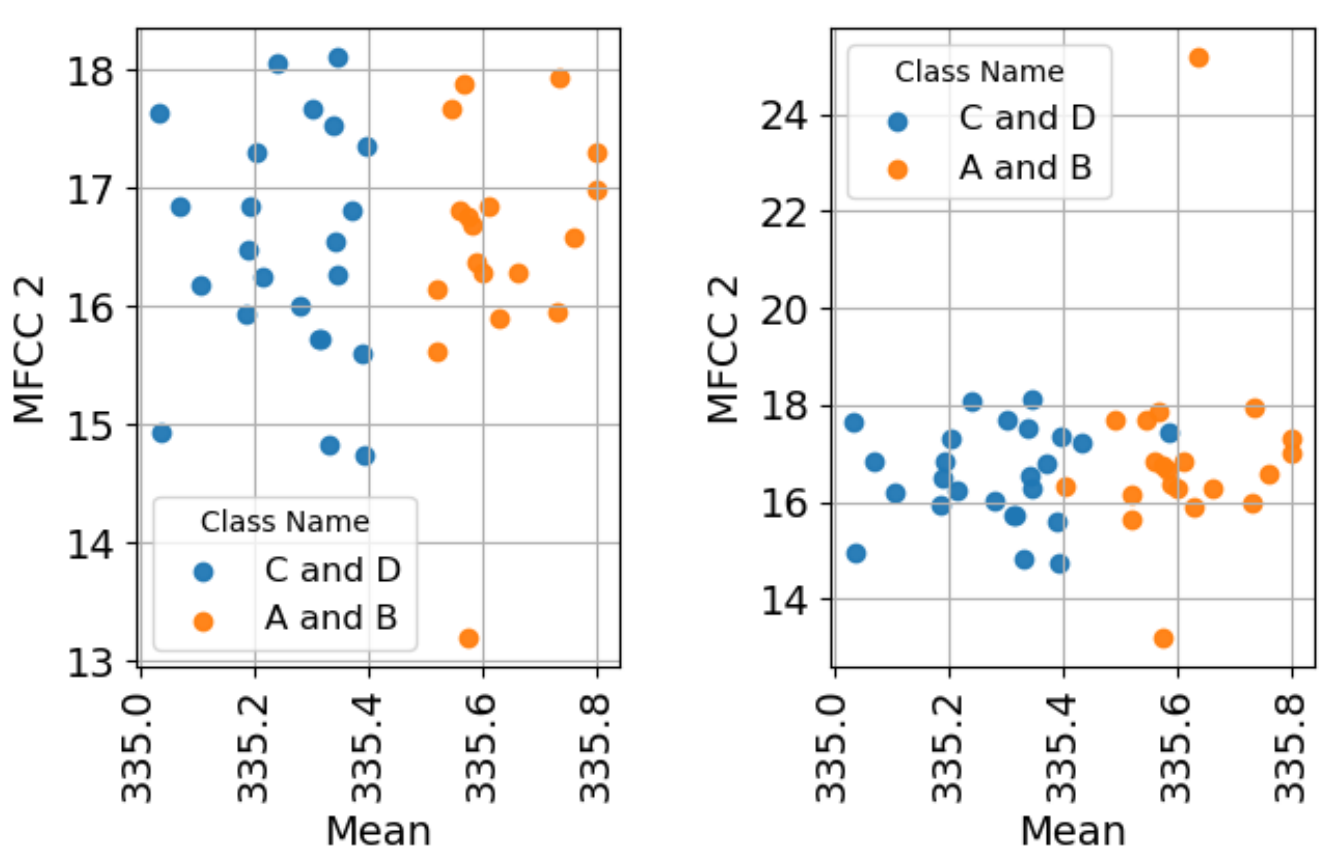

(a) HDMI

(b) USB

Fig. 6: Plots of Mean feature and MFCC 2 for Matrix Multiplication experiments.

The experimental procedure begins with vigilant I/O port monitoring, followed by the beginning of CNN classification using the image data sets delineated in V-C. To ensure meticulous data integrity, a deliberate interval of 10 seconds is observed between each data collection sample. Recognizing the potential utilization of alternative CNN models by the victim, our experimental framework is also equipped to detect the presence of any unidentified model classes. We meticulously crafted our experimental design into two distinct configurations, Configuration 1 and Configuration 2, to probe the depths of CNN architecture inference. In Configuration 1, the attacker is armed with prior knowledge of the target CNN model's power consumption patterns, which are included in his training dataset. This setup allows for a direct and informed inference of the CNN model. Moving to Configuration 2, we introduced a layer of complexity where the attacker is only aware of the power data for one variant of the CNN model. This strategic limitation challenges the attacker to deduce the specific family to which an unknown variant of the CNN model belongs, based on partial data. This approach enhances our understanding of how well an attacker can generalize from known models to detect novel yet related model architectures. The data sets used in this experiment, which serve as input to CNN, are thoroughly described in Section V-C1.

2) Inference of Videos Rendered on the GPU: Now a days, creating videos for the purpose of entertainment, video blogging (vlogging), or education is a very common task. One of the major use cases of the GPU of a computer is rendering videos. We further conduct an experiment evolving around video rendering. We design this experiment where the attacker wants to identify some video which is being rendered on the victim's machine. To run the experiment programmatically, we utilized the OpenCV library [38] with Python. Here we use 10 different videos from different sources and try to classify the video. We assume that the attacker can train machine learning models with power leakage data for different videos. If such a video is rendered on the victim machine, the attacker can identify that by observing the power leakage data over the HDMI port of the victim machine. In this experiment, each

video is cut for 10 seconds after the video is played for 10 seconds. We repeat this process for all the videos and 25 times each.

C. Experiment Datasets

In the following section, we present a detailed overview of the datasets utilized in our experiments. We describe the characteristics of these datasets, emphasizing their relevance for our experimental objectives.

1) Image Datasets for testing CNNs: The Convolutional Neural Networks (CNNs) typically take an image as input to perform different tasks (such as classification). This section discusses the image datasets used in our experiments for the CNNs. The images were taken from the ImageNet dataset [39]. The ImageNet dataset is widely used in the ImageNet competitions, which has produced several CNNs over the years, such as VGG, ResNet, etc. We used 2 different categories of images. Dataset 1 contains the images of 50 cats and 50 dogs. Cats and dogs are typically misclassified among themselves. Dataset-2 contains 100 random images. We have further divided the data sets into two parts. In the later part of our experiment, we classify the CNN architectures – inferred from the collected Power leakage. We had 80 images selected for the train set and 20 images for the test set. There were no common images between the train and the test sets.

2) Video Inference Experiment: In this experiment we need to use video files as input to video rendering experiment. In the process of determining the appropriate format and quality of the video utilized in the Video Identification Experiment, the ubiquity of smartphones as recording devices was considered. Contemporary smartphones generally possess the capability to record in Full HD (1080p) or superior resolutions. The mp4 file format, commonly employed for the storage of video content, coupled with the H.264 video codec, strikes an optimal balance between superior video quality and minimized bandwidth consumption, rendering it exemplary for streaming and broadcasting applications. Consequently, the H.264 codec was selected for post-editing video rendering. The corpus of video files chosen for this experiment comprised ten videos, each with a duration ranging from a minimum of 30 seconds to a maximum of one minute.

VI. Evaluation

In this section, we present a comprehensive evaluation of our experiments. The evaluation is divided into two primary segments: Preliminary Data Exploration (Section VI-A) and Classification Results (Section VI-B). This evaluation also presents reliable accuracy of the attack experiments and features impacting the experiments of CNN model architecture inference and the video inference used in video rendering tasks running on the GPU for both I/O ports (USB and HDMI).

A. Preliminary Data exploration

In this study segment, we discuss how the raw data was prepared with feature extraction and removal of empty entries for classification. Initially, the data were captured in analog values form during the experiments. Afterward, each sample was stored as an array and subsequently organized into a dataframe. Missing or NaN values were removed after the data collection. In order to identify all CNN models from the collected power leakage data, we employed statistical and spectral feature extraction methods using Python's NumPy [40] and Librosa [41], respectively. The extracted features, used in our experiments, both statistical and spectral, are outlined in Table II.

Feature Type	**Extracted Features**
Statistical Features	Summation, Mean, Mean Absolute Deviation (MAD),Standard Deviation, Variance, Skewness, Standard Error of the Mean (SEM), Kurtosis
Spectral Features	Spectral Centroids (Multiple), Spectral Bandwidths (Multiple), Spectral Flatnesses (Multiple), Spectral Roll-offs (Multiple), Tempograms (Multiple), Spectral Contrasts (Multiple), Fourier Tempograms (Multiple), Zero Crossing Rate, Mel-frequency Cepstral Coefficients (Multiple), Tempo, Chroma (Multiple), Tempogram Ratio, Root Mean Square (RMS)

TABLE II: Statistical and Spectral features that were extracted from the collected data.

After the feature extraction process, we investigated the features that were extracted. For this part, we considered the features extracted using the Dataset-2 for visualization purposes. In order to identify the best features, we used the Scikit learn's [42] Select K-Best method for the top 30 features. First, we discuss the CNN models identification feature extraction process. We found that the feature sum and roll-off (1st roll-off) are of the highest importance in terms of both statistical and spectral features. It is important to mention that many spectral features produce more than one value for that specific based on the characteristics of the given time series. We also observed that chroma, flatness, SEM, and RMS are of higher importance. We presented the importance value created by the Select K-Best for the top 30 features in Figure 7. In addition, we investigated the values of some of the top features. In Figure 8, we present the two main features - Sum (Statistical feature) and Chroma 9 (Spectral feature). This figure presents box plots of the values for all CNNs involved in the experiment for those two features. We can observe that some of the CNN models are distinguishable visually by observing these two features.

Afterwards, we discuss the feature extraction outcome of the video identification experiment. We found the top features to be Sum, Chroma 9, Flatness 0, Chroma 11 and Rolloff features. In this part, we visualize the values of two spectral features, Chroma 9 and Flatness 0. The values of these two features are presented in figure 9. It can be seen that Chroma 9 and Flatness 0 have visually distinguishable features for the 10 different videos.

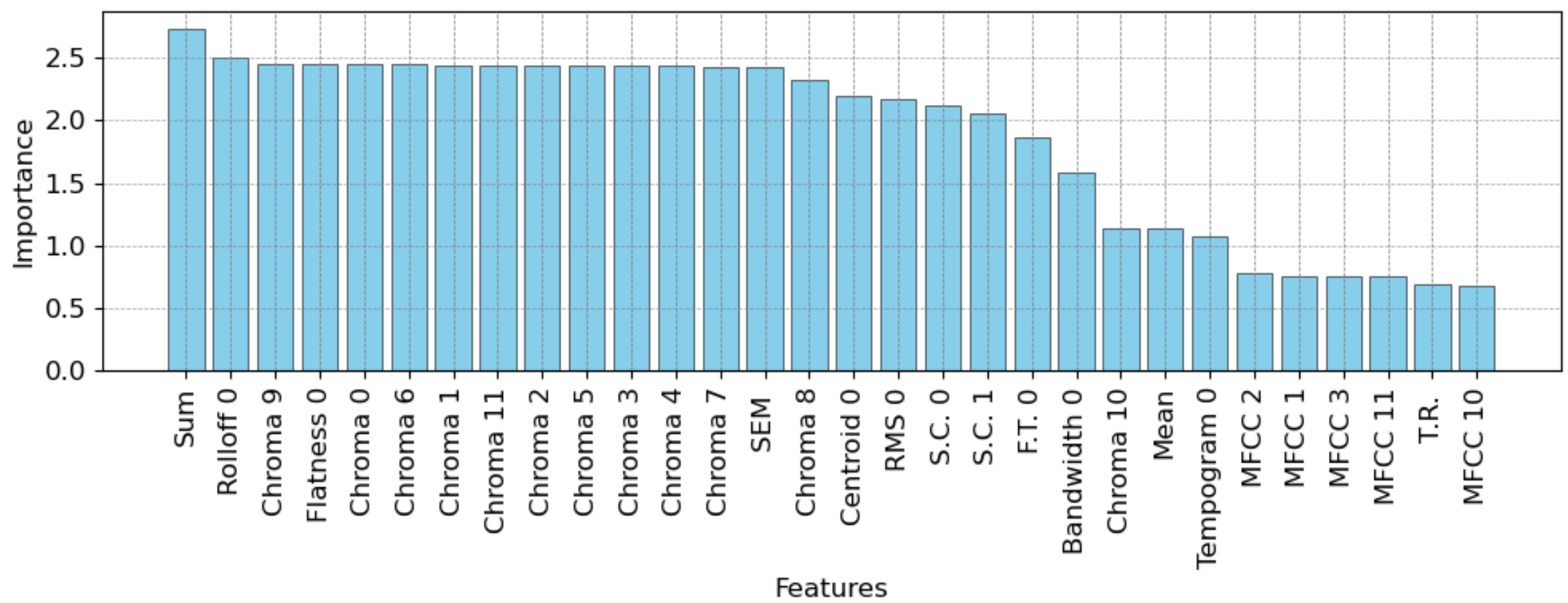

Fig. 7: Feature importance from Scikit learn's Select K-Best for the top 30 features, derived from CNN models on Dataset-2. Abbreviations: F.T. (Fourier Tempogram), T.R. (Tempogram Ratio), S.C. (Spectral Contrast).

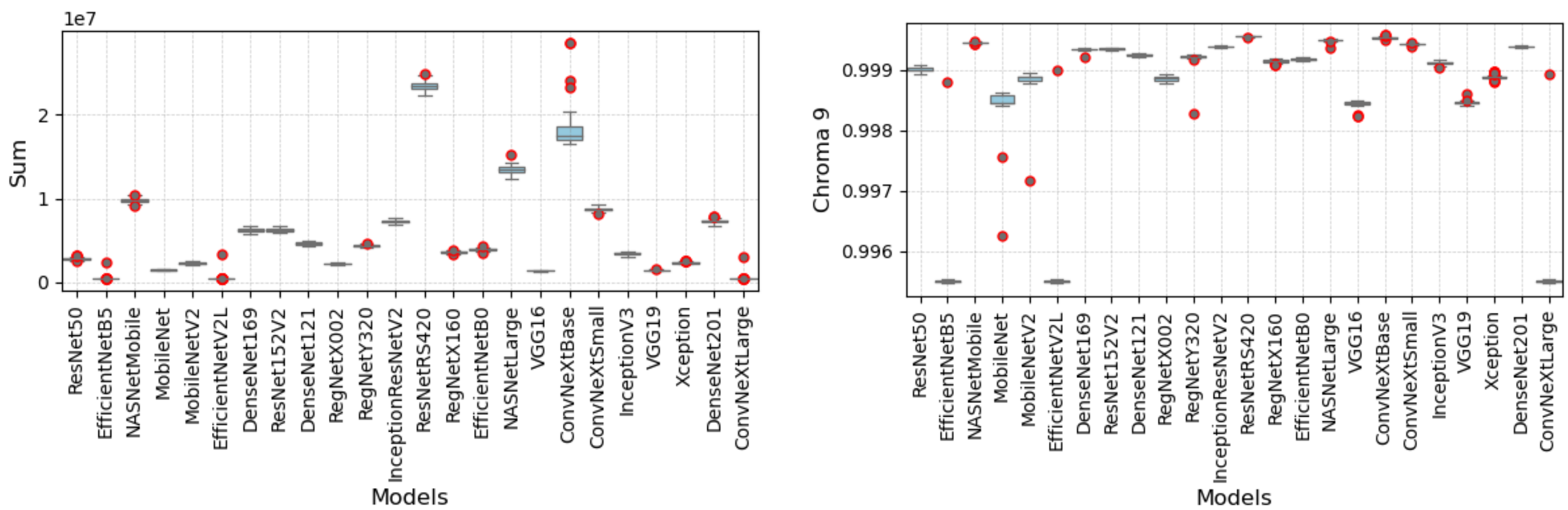

(a) Feature: Sum

(b) Feature: Chroma 9

Fig. 8: Box plots presenting the top two features - Sum (Statistical feature) and Chroma 9 (Spectral feature) for all the CNN models involved in the experiments. The red dots represents the outliers.

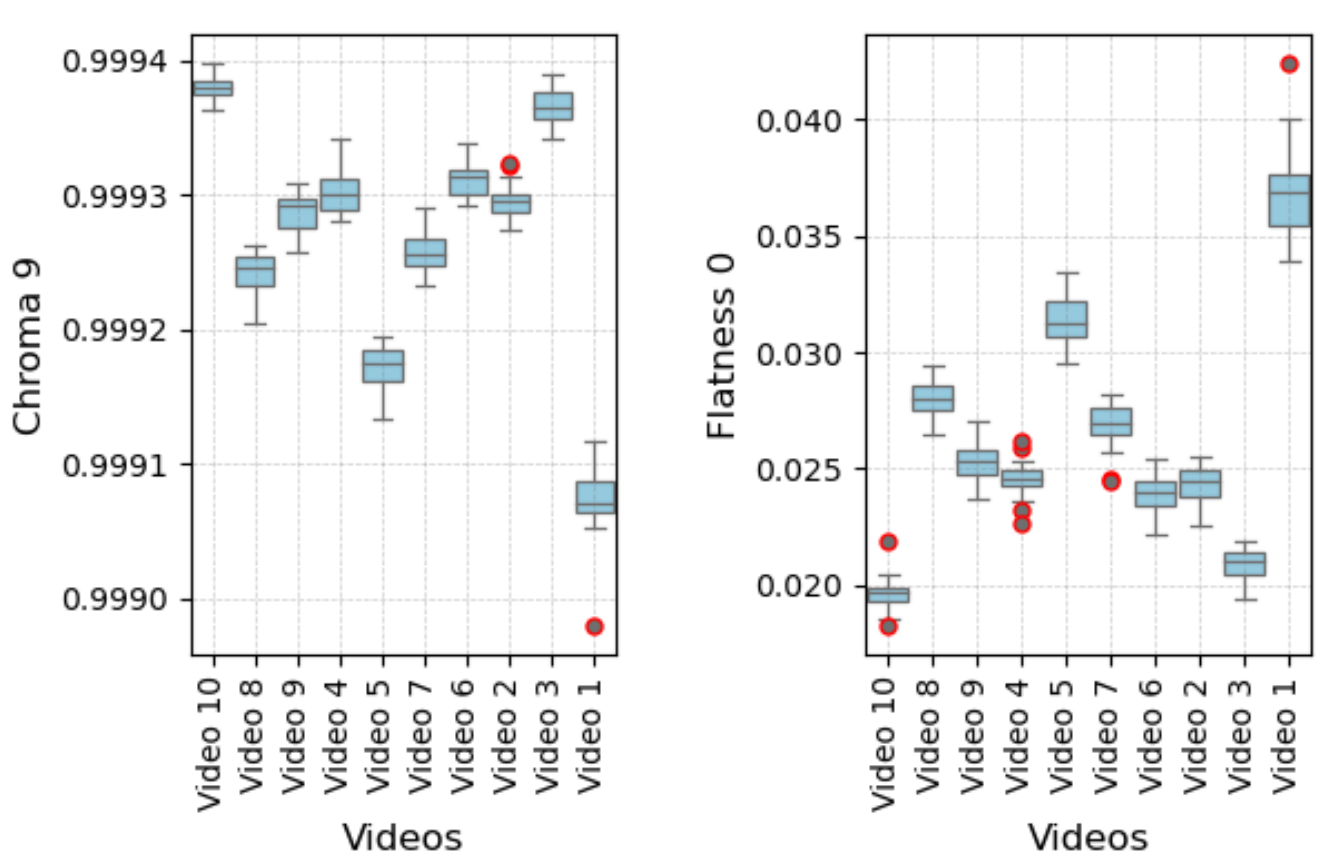

(a) Chroma 9

(b) Flatness 0

Fig. 9: Plots of Mean feature and MFCC 2 for Matrix Multiplication experiments.

B. Classification

In our research, we utilized the Scikit-learn library [42] to develop pipelines using various scalars and classifiers for data preprocessing and classification tasks, respectively. The scalars included: Max-Abs Scaler, ensuring the maximum absolute value of each feature is 1.0 without centering; Min-Max Scaler, adjusting features to the minimum and maximum range; Standard Scaler, normalizing features by removing the mean and scaling to unit variance; Quantile Transformer, mapping data to a uniform distribution and minimizing outlier effects; and Normalizer, scaling samples to unit norm, suitable for datasets with many zeros and varying scales. Classifiers employed were K-Neighbors [43], Light GBM [44], Support Vector Machine (Multiclass) [45], Random Forest [46], and XGBoost [47]. We used the K-best algorithm to select the top 30 features for the pipelines, choosing the classifier with the highest test accuracy and the F1 score. We also set a confidence threshold of 0.5 for all classifiers to handle unknown class cases. Next, we discuss the classification results for the CNN and Video identification

experiments in Section VI-B1 and VI-B2.

1) Classification results for CNN model architecture inference experiment: We first identify the base models of the networks mentioned in Table I with multiclass classification. As mentioned before, we execute the sckit-learn pipeline composed of the scalar and the classifier. We run the classification step on both data sets (datasets 1 and 2) independently. This is formulated to be a 10-class problem. We conducted the classification task with the possibility of an unknown class and also without the possibility of having an unknown class. The test accuracy and the F1 score for the best performing model are given in Table III. We can observe that, while monitoring the HDMI port, Dataset-2 had performed well with an accuracy of 94.25% for the regular classification. Considering having an unknown class, the Dataset-2 performed better than the Dataset-1, having a test accuracy of 93.75%. In both cases, the F1 score is higher. Furthermore, we present a confusion matrix for the Dataset-2 classification in Figure 10. We can observe that in some of the test cases, MobileNet was misclassified as VGG16, InceptionNet was misclassified as MobileNet, and RegNet was misclassified as EfficientNet. Moreover, upon meticulous examination of the USB port, Dataset-2 exhibited a remarkable accuracy of 98.95% when accounting for the potential presence of an unknown class. In contrast, when the scenario excluded the consideration of an unknown class, Dataset-1 demonstrated superior test accuracy, achieving an impressive 71. 35%.

Subsequently, we executed the classification for Configuration 2. It should be noted that, during the HDMI port monitoring, Dataset-2 exhibited commendable performance, achieving an accuracy of 76.45% in standard classification scenarios. When incorporating an unknown class into the analysis (by setting the confidence threshold to 0.5 and providing unknown power leakage data), Dataset-2 outperformed Dataset-1, registering a test accuracy of 74.39%. In both cases, the F1 scores were significantly higher, underscoring the robustness of the model. Furthermore, a thorough investigation of the USB port revealed that Dataset-2 maintained a remarkable accuracy of 74.39% while considering the potential inclusion of an unknown class. In contrast, in scenarios where the unknown class was not considered, Dataset-2 showed superior test accuracy, attaining an impressive 72.10%. The results are presented in Table IV.

Across all instances examined, the K-Neighbors classifier emerged as the superior model, employing data normalization via the sklearn normalizer scaler. This model was meticulously configured through a pipeline designed to fine-tune hyperparameters. The optimized parameters included: the algorithm parameter was configured to ’auto’ to enable automatic selection of the algorithm, leaf_size was established at 30, the metric was designated as Minkowski with p=2 to denote Euclidean distance, and n_neighbors was fixed at 5 with weights set to uniform. However, in some of the cases the XGBoost classifier performed well, such as Dataset-2, unknown class classification while monitoring the USB port. The random forest classifier performed better than the other classifiers when working with Dataset-1 and for regular classification (without considering the possibility of unknown class) while monitoring the USB port.

Classification	Dataset	HDMI		USB	
		Accuracy	F1 Score	Accuracy	F1 Score
Regular	Dataset-1	87.36%	0.8747	71.35%	0.7063
	Dataset-2	94.25%	0.9415	56.77%	0.5661
Unknown	Dataset-1	86.25%	0.8638	98.34%	0.9814
	Dataset-2	93.75%	0.9364	98.85%	0.9885

TABLE III: Test accuracy for the Base CNNs classification (Configuration 1).

Classification	Dataset	HDMI		USB	
		Accuracy	F1 Score	Accuracy	F1 Score
Regular	Dataset-1	68.34%	0.6829	68.48%	0.6833
	Dataset-2	76.45%	0.7667	72.10%	0.7187
Unknown	Dataset-1	62.58%	0.6256	60.50%	0.6014
	Dataset-2	74.39%	0.7437	71.39%	0.7321

TABLE IV: Test accuracy for the CNN family classification (Configuration 2).

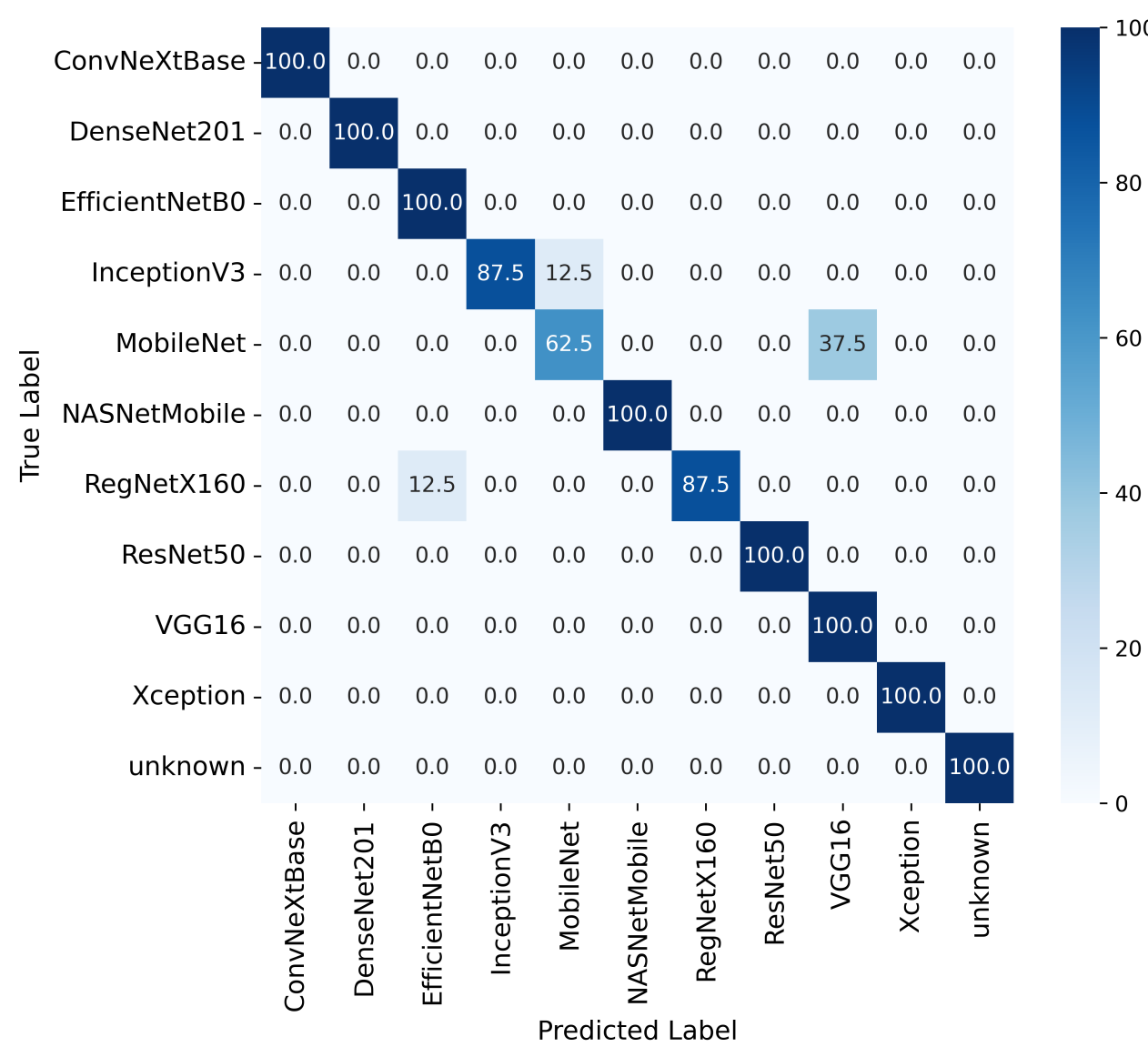

Fig. 10: Confusion Matrix for the Classification of Base CNN models (Configuration 1), for the Dataset-2.

2) Classification results for video rendering inference experiment: For the video identification task, we again perform a multiclass classification. We use the same machine learning pipelines discussed earlier in this classification task. In this experiment, the K neighbor classifier with the normalizer scalar also performed the best.

The test accuracy obtained in this classification task with the USB port is 92.86%. The F1 score is 0.92. The scikit learn pipeline optimizes the classification task by setting the hyperparameters of the classifiers. In this experiment, the hyper parameters of the K neighbor classifier were set as follows. The leaf_size is set to 30, metric as Minkowski with p = 2 representing the Euclidean distance, and n_neighbors set to 5 with weights as uniform. We also present a confusion matrix for this experiment in Figure 11. We can observe that video 2 and video 4 were misclassified among themselves for a few test cases. On the other hand, with the identical classifier

Classification	USB		HDMI	
	Accuracy	F1 Score	Accuracy	F1 Score
Regular	92.86%	0.92	100%	1
Unknown	96.00%	0.9589	96.43%	0.9609

TABLE V: Classification results for the video identification experiment

configuration, the test accuracy for the HDMI port achieved an impeccable 100%. When we considered the possibility of having an unknown class by setting the confidence threshold to 0.5 and providing unknown power leakage data, the best performing average test accuracy was observed from Dataset-2 on both HDMI and USB ports, which were 74.39% and 71.39% respectively. The sklearn classification configuration remained same as before in this case.

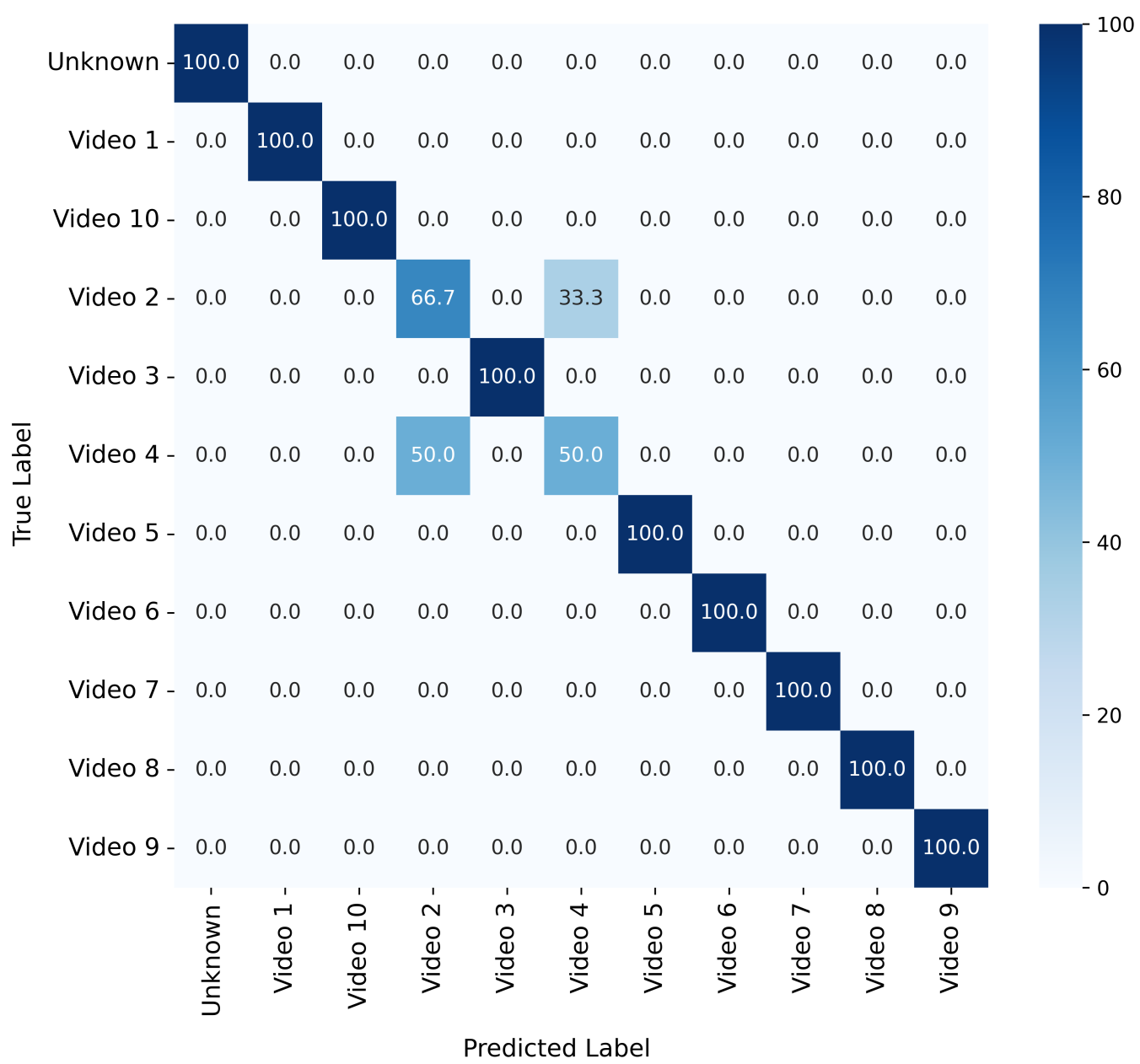

Fig. 11: Confusion Matrix for the video rendering inference experiment.

VII. Limitation

In this section, we delve into the critical limitations of the power side-channel, which can significantly influence the accuracy of the attack. We explore two pivotal limitations here: first, the necessity for the proximity of the attacker, where the attacker must be in close proximity to the victim (Section VII-A). Secondly, if the computer is executing multiple programs simultaneously, the attack may become infeasible (Section VII-B). Hence, the consideration of background noise is paramount.

A. Attacker Proximity

Our attack necessitates the attacker to meticulously read the current readings from the I/O port of the victim. This can be accomplished through either a USB cable directly linking the attack device or via wireless communication. Our sophisticated attack device can be seamlessly extended to utilize the WiFi or Bluetooth module, replacing the traditional wired USB connection between the victim and the attacker. Leveraging wireless communication empowers the attacker to significantly increase the distance from the victim, thereby minimizing the risk of detection. However, the maximum feasible distance is constrained by the coverage range of WiFi or Bluetooth technology. Notably, certain Bluetooth 5.3 version products boast an impressive effective range of up to 25 feet. In contrast, WiFi modules can span distances of up to 120 feet. Consequently, the attacker must be in close proximity to the victim to gather power side-channel data, necessitating a certain degree of nearness to the victim.

On another note, the attacker could modify the attack device to store the power side channel data from the victim onto a temporary storage medium, such as an SD card. This allows the attacker to later retrieve the SD card from the compromised device connected to the victim's machine. In such a scenario, the attacker must be physically present near the victim's machine not only to connect the attack device initially, but also to return and retrieve the collected data. This dual presence heightens the risk and audacity of the attack, showcasing the lengths to which an attacker might go to exploit vulnerabilities.

B. Background Noise Consideration

While conducting our experiments, we deliberately chose not to disable any system processes. We ensured that all operating system-related processes remained active. To maintain a controlled environment, we meticulously avoided any additional services or software from the user end, such as web browsers, games, etc. However, theoretically speaking, the presence of other running processes on the system could potentially interfere with or distort the attack. We rigorously tested several fundamental operations, including matrix multiplication, video rendering, and CNN execution. If a similar process were to run concurrently, it could indeed weaken the attack. Consequently, background noise can significantly impact the attack.

VIII. Conclusion

This paper unveils an innovative methodology for executing power side-channel attacks by redirecting the emphasis from the conventional practice of directly measuring the power consumption of processing units to scrutinizing peripheral components that share the same power supply. In particular, we explore the potential for harnessing power fluctuations in USB and HDMI ports as a novel vector for such attacks. This innovative mechanism can be used adeptly to extract valuable information from the I/O ports of aging computational devices. This opens new avenues for security research and exploitation.

We meticulously crafted a series of experiments to investigate the complicated operations of GPUs. Our initial focus was on unraveling the unique power leakage characteristics during simple matrix multiplication on a GPU. We found that by scrutinizing the power leakage, one can extract valuable insights about the matrix multiplication processes executed

on the GPU. Building upon this revelation, we directed our research by orchestrating a comprehensive array of experiments. These experiments spanned diverse tasks, including the intricate realms of video rendering and the sophisticated architectures of convolutional neural networks (CNNs), to meticulously infer the underlying architecture of CNN models.

We structured our experiments to identify the family of CNN models that are used on the GPU and further tried to identify the variant of the CNN model that is used on the GPU. Additionally, we explored the potential for identifying hitherto unknown classes within our experimental framework. In our experiments, we investigated two widely used I/O ports, such as USB and HDMI ports in all our experiments.

Our findings demonstrate that power fluctuations in peripheral components can indeed be harnessed to reveal significant information about GPU operations such as video rendering, matrix multiplication, and the architecture of neural network models. This not only broadens the scope of power side-channel attacks but also underscores the need for comprehensive security measures that encompass all aspects of the power supply network. By shifting the focus from direct power measurement of processing units to peripheral components, we provide a new perspective on the potential vectors for power side-channel attacks.

The power leakages enabling our attack could be due to several reasons. First of all, computer power supply units (PSU) stabilize the electrical current of system components using capacitors, MOSFETs and voltage regulation modules (VRMs). These components ensure consistent current flow, protecting against performance or power fluctuations [48]. However, due to wear and tear the effectiveness of power supply stabilization may deteriorate over time. This could contribute to the power draw patterns that our attack picks up at the USB and HDMI ports. Another possible reason could be that the power stabilization mechanisms of today's computers are simply not fine-grained enough to obfuscate the minute power fluctuation patterns that our algorithms leverage to drive the attacks. A thorough exploration on the potential contributions of these variables is part of our future work.

The implications of this study offer new insights and opportunities for both attackers and defenders in the ongoing battle to secure computational devices.

References

[1] N. Fox, "How to use autoruns to detect and remove malware on windows." Varonis, 2022. [Online]. Available: https://www.varonis.com/blog/how-to-use-autoruns

[2] C. Levi, "Guide to computer ports and connectors: Usb, thunderbolt 3, hdmi." Hardware Corner, 2020. [Online]. Available: https://www.hardware-corner.net/guides/guide-to-computer-ports-and-connectors

[3] A. Piltch, "A guide to computer ports and adapters." Laptop Mag, 2022. [Online]. Available: https://www.laptopmag.com/articles/port-and-adapter-guide

[4] "Nvidia system management interface," https://developer.nvidia.com/nvidia-system-management-interface/, accessed: June 2, 2024.

[5] "Nvidia Management Library," https://developer.nvidia.com/nvidia-management-library-nvml/, accessed: June 2, 2024.

[6] N. K. Jha, S. Mittal, B. Kumar, and G. Mattela, "Deeppeep: Exploiting design ramifications to decipher the architecture of compact dnns," *J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 1, oct 2020. [Online]. Available: https://doi.org/10.1145/3414552

[7] M. Sabbagh, Y. Fei, and D. Kaeli, "A novel gpu overdrive fault attack," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[8] S. E. Arefin and A. Serwadda, "Dissecting the origins of the power side-channel in deep neural networks," in *2024 IEEE 5th Annual World AI and IoT Congress*, 2024.

[9] M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss, "Platypus: Software-based power side-channel attacks on x86," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 355–371.

[10] X. Zhang, A. A. Ding, and Y. Fei, "Deep-learning model extraction through software-based power side-channel," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[11] R. Segev and A. Mendelson, "The use of performance-countersto perform side-channel attacks," in *Cyber Security, Cryptology, and Machine Learning: 7th International Symposium, CSCML 2023, Be'er Sheva, Israel, June 29–30, 2023, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2023, p. 216–233. [Online]. Available: https://doi.org/10.1007/978-3-031-34671-2_16

[12] C. Liu, A. Chakraborty, N. Chawla, and N. Roggel, "Frequency throttling side-channel attack," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. ACM, Nov. 2022. [Online]. Available: http://dx.doi.org/10.1145/3548606.3560682

[13] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, and D. Kohlbrenner, "Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86," *IEEE Micro*, vol. 43, no. 4, pp. 19–27, 2023.

[14] D. R. Dipta and B. Gulmezoglu, "Df-sca: Dynamic frequency side channel attacks are practical," in *Proceedings of the 38th Annual Computer Security Applications Conference*, ser. ACSAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 841–853. [Online]. Available: https://doi.org/10.1145/3564625.3567979

[15] N. Sehatbakhsh, B. B. Yilmaz, A. Zajic, and M. Prvulovic, "A new side-channel vulnerability on modern computers by exploiting electromagnetic emanations from the power management unit," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 123–138.

[16] H. Gamaarachchi and H. Ganegoda, "Power analysis based side channel attack," 2018.

[17] C. Luo, Y. Fei, P. Luo, S. Mukherjee, and D. Kaeli, "Side-channel power analysis of a gpu aes implementation," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, 2015, pp. 281–288.

[18] Q. Yang, P. Gasti, G. Zhou, A. Farajidavar, and K. S. Balagani, "On inferring browsing activity on smartphones via usb power analysis side-channel," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1056–1066, 2017.

[19] S. Acharya, A. Serwadda, and A. V. Bilbao, "That phone charging hub knows your video playlist!" in *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, 2021, pp. 160–169.

[20] A. S. La Cour, K. K. Afridi, and G. E. Suh, "Wireless charging power side-channel attacks," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 651–665. [Online]. Available: https://doi.org/10.1145/3460120.3484733

[21] M. Zhao and G. E. Suh, "Fpga-based remote power side-channel attacks," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 229–244.

[22] S. E. Arefin and A. Serwadda, "Deep neural exposure: You can run, but not hide your neural network architecture!" ser. IH&MMSec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 75–80. [Online]. Available: https://doi.org/10.1145/3437880.3460415

[23] A. D. Ramadhanty, A. Budiono, and A. Almaarif, "Implementation and analysis of keyboard injection attack using usb devices in windows operating system," in *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, 2020, pp. 449–454.

[24] D. V. Pham, A. Syed, and M. N. Halgamuge, "Universal serial bus based software attacks and protection solutions," *Digital Investigation*, vol. 7, no. 3, pp. 172–184, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287611000041

[25] K. Nohl, S. Krißler, and J. Lell, "Badusb—on accessories that turn evil," in *BlackHat USA 2014*. Security Research Labs, 2014. [Online]. Available: https://www.blackhat.com/us-14/briefings.html#Nohl

[26] H. Jeong, Y. Choi, W. Jeon, F. Yang, Y. Lee, S. Kim, and D. Won, "Vulnerability analysis of secure usb flash drives," in *2007 IEEE International Workshop on Memory Technology, Design and Testing*, 2007, pp. 61–64.

[27] B. Anderson and B. Anderson, *Seven deadliest USB attacks*. Syngress, 2010.

[28] Kaspersky, "Equation group: Questions and answers. 2015. url: https://securelist. com/files/2015/02," *Equation _ group_questions_and_answers. pdf*.

[29] U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat, "A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review," *Sensors (Basel)*, vol. 23, no. 8, p. 4117, Apr 2023.

[30] K. Satrinekarn, "Black 850w power supply unit with cables and switch i/o for atx tower pc cases," https://www.vecteezy.com/photo/2221759-black-850w-power-supply-unit-with-cables-and-switch-i-o-for-atx-tower-pc-cases-isolated-on-a-white-background, 2023, accessed: April 8, 2024.

[31] "Power supply connectors - how to connect pc power connectors," https://pcfastlane.com/power-supply-connectors-guide-understanding-types-and-functions/, 2023, accessed: April 4, 2024.

[32] SparkFun Electronics, "Sparkfun qwiic shield for arduino nano," https://www.sparkfun.com/products/14544, 2023, accessed: June 2, 2024.

[33] "Arduino MKR Zero," url: https://docs.arduino.cc/hardware/mkr-zero/, accessed: January 2, 2024.

[34] Glmark2 Development Team, "glmark2," https://github.com/glmark2/glmark2, 2024, accessed: March 20, 2024.

[35] NVTop Contributors, "Nvtop: Nvidia gpus monitoring tool," https://github.com/Syllo/nvtop, 2024, accessed: March 20, 2024.

[36] S. Tokui *et al.*, "Cupy: A numpy-compatible array library for gpu-accelerated computing with python," https://cupy.dev/, 2023, accessed: March 21, 2024.

[37] "TensorFlow module: tf.keras.applications," Website, 2024, tensorFlow version: v2.12. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/applications

[38] G. Bradski, A. Kaehler *et al.*, "OpenCV: Open source computer vision library," Software available from http://opencv.org, 2023, accessed: March 20, 2024.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[40] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[41] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[43] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, R. Meersman, Z. Tari, and D. C. Schmidt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 986–996.

[44] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: a highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 3149–3157.

[45] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[46] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 10 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[47] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[48] "About vrms & mosfets / motherboard safety with high-tdp processors," https://www.overclock.net/threads/about-vrms-mosfets-motherboard-safety-with-high-tdp-processors.1772883/, 2011, accessed: April 4, 2024.