ERSP
UC San Diego
Early Research Scholars Program

RLBox

Sandboxing libjpeg in TensorFlow

Lauren Chen, Michael Granado, Anokhi Mehta, and Jenelle Truong

Advisors: Shravan Narayan and Deian Stefan

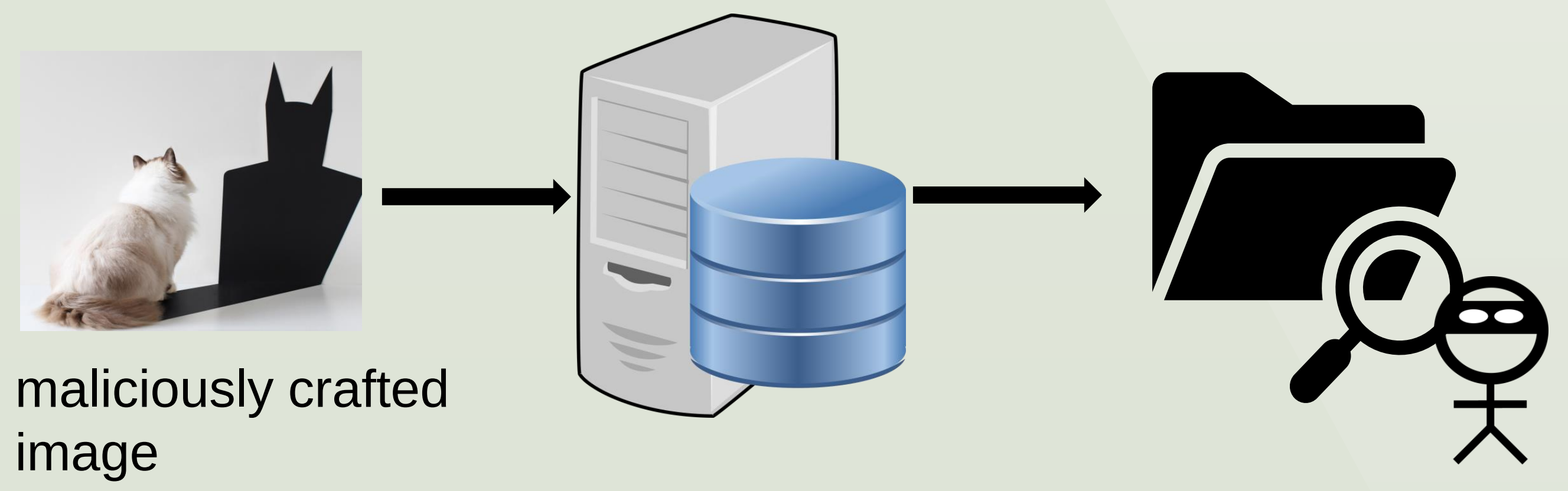
UC San Diego
JACOBS SCHOOL OF ENGINEERING
Computer Science and Engineering

MOTIVATION

To render different types of media content, like videos and images, applications rely on third-party libraries. However, these third-party libraries are often written in memory-unsafe languages. In effect, attackers can exploit memory-related vulnerabilities in these libraries to compromise application servers and gain access to sensitive information.

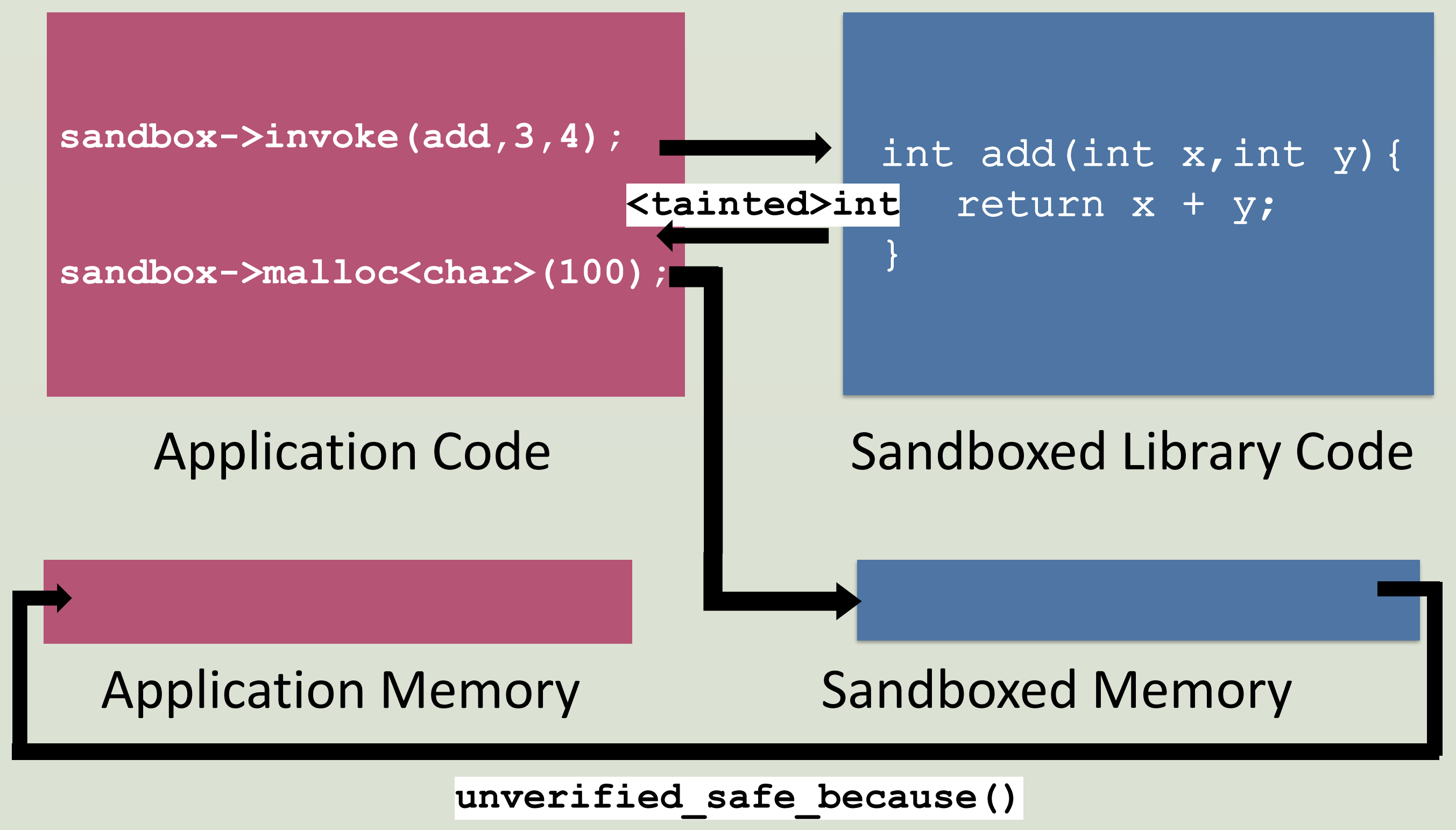
RESEARCH PROBLEM

TensorFlow renders images through the third-party C library `libjpeg`. By feeding a corrupted JPEG into an application that runs TensorFlow on its servers, attackers can then access sensitive server information.



SOLUTION

By separating code execution from memory, sandboxes prevent the compromise of the application servers.



```
sandbox->invoke(add,3,4);
sandbox->malloc<char>(100);
```

```
int add(int x,int y){
  return x + y;
}
```

`<tainted>int`

`unverified_safe_because()`

Figure 1: Memory isolation using sandboxes

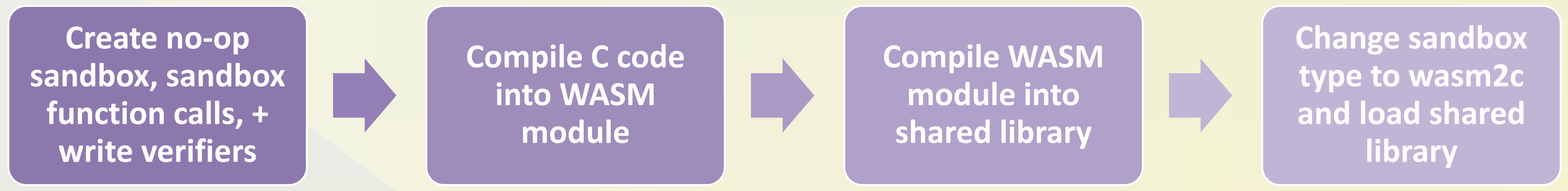
Research question:
Can we secure TensorFlow from malicious JPEG images by sandboxing `libjpeg` while keeping performance overheads to a minimum?

IMPLEMENTATION

RLBox is a framework that:

- Takes a fine-grained approach to isolate libraries in their own sandboxes
- Utilizes sandboxes to prevent threats from exploiting vulnerabilities and compromising an entire application (i.e bypassing bounds checks to read data outside of an array)

Using RLBox, we followed the workflow presented here:


Create no-op sandbox, sandbox function calls, + write verifiers
Compile C code into WASM module
Compile WASM module into shared library
Change sandbox type to wasm2c and load shared library

No-op sandboxes of type `rlbox_noop_sandbox` do not perform memory isolation but allow for an easier porting process. After ensuring correct implementation of the RLBox framework, we can change the sandbox type to `rlbox_wasm2c_sandbox`, which is WebAssembly (WASM) based and performs memory isolation.

The example below demonstrates the process of implementing the RLBox framework to `libjpeg` within TensorFlow:

Unsandboxed Code

```
struct jpeg_decompress_struct cinfo;
struct jpeg_error_mgr jerr;
cinfo.err = jpeg_std_error(&jerr);
```

No-op Implementation

```
rlbox_sandbox<rlbox::rlbox_noop_sandbox> sandbox;
sandbox.create_sandbox();
…
auto p_cinfo = sandbox.malloc_in_sandbox<jpeg_decompress_struct>
               (sizeof(jpeg_decompress_struct));
auto p_jerr = sandbox.malloc_in_sandbox<jpeg_error_mgr>
              (sizeof(jpeg_error_mgr));
p_cinfo->err = sandbox.invoke_sandbox_function(jpeg_std_error,
               p_jerr);
```

Figure 2: Example comparing portions of the UncompressLow() method from TensorFlow across the unsandboxed and no-op implementation

The no-op implementation allocates space in sandboxed memory for the `jpeg_decompress_struct` and `jpeg_error_mgr` before invoking a function with these two tainted types, whereas the unsandboxed code does all of this in application memory.

RESULTS AND EVALUATION

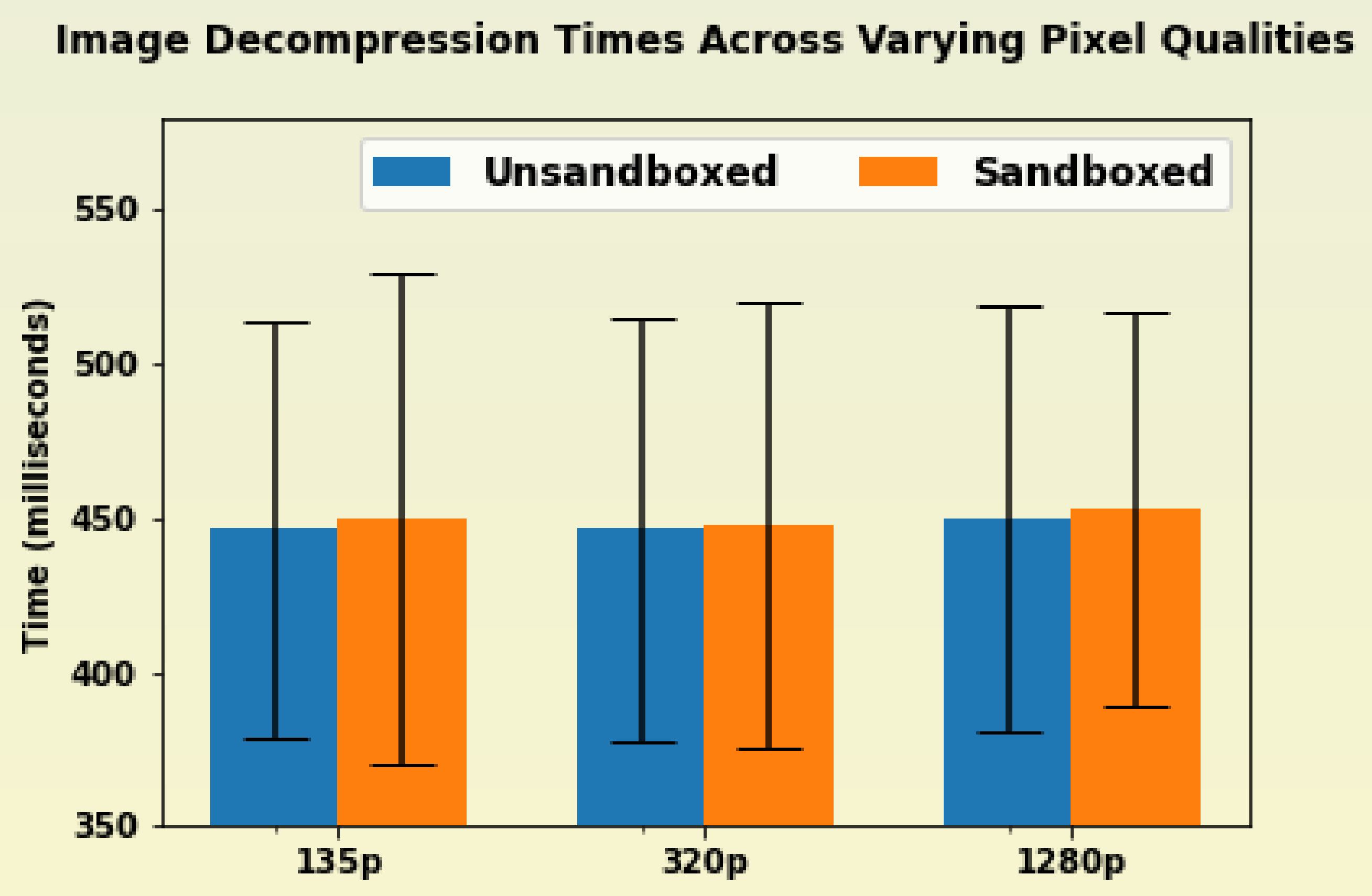


Figure 3: Bar graph comparing image decompression times across varying pixel qualities for the unsandboxed and sandboxed versions of `libjpeg` in TensorFlow

To evaluate any performance overheads caused by the sandbox mechanism, we compare the unsandboxed and sandboxed implementations of TensorFlow on an image classification model that decompresses images. Our results indicate that there are no significant overheads, meaning that we can ensure the secure handling of JPEG images without sacrificing performance.

CONCLUSION AND FUTURE WORK

Based on our results, we can conclude that implementing the RLBox framework to `libjpeg` in TensorFlow will ensure security while minimizing performance overheads.

To take our project a step further, we can sandbox other third-party libraries within TensorFlow. We can also sandbox the `libjpeg` library within applications other than TensorFlow.

ACKNOWLEDGEMENTS

We would like to thank the ERSP program, Christine Alvarado, Mai ElSherief, Vaidehi Gupta, Shravan Narayan, and Deian Stefan.