Concrete-ML FHE SARS-CoV-2 Strain Classifier User Manual

Johann Benjamin Vivas

July 2023

1 Getting Started

The Concrete-ML FHE SARS-CoV-2 Strain Classifier is a client-server application that allows clients to hash and encrypt SARS-CoV-2 viral sequences, and send these encrypted sequences over to the server-side application of the system for classification via its API. The system was developed on WSL2 using the Django Framework for the server-side application, and CustomTkinter and Tkinter for the client-side GUI application, with the Concrete-ML FHE ML library providing the main encryption, prediction, and decryption features of the system.

1.1 System Requirements

The requirements for running the system at full capacity include the following specifications:

- **Operating System**: Linux (x86) or Windows Subsystem for Linux 2
- **Memory**: 4GB minimum
- **Storage space**: 7GB free disk space for Concrete-ML package and dependencies
- **Network interface**: Ethernet or Wi-Fi

An estimated 7 gigabytes of free disk space is required to run the full system due to `Concrete-ML`'s dependencies, including `torch` and `concrete`. Additionally, as the client application requires the Dashing tool, the appropriate shell scripts, a text file listing the selected features for classification and original class labels (`features_and_classes.txt`), and the `client.zip` file, which holds the required cryptographic parameters for key generation, encryption, and decryption, the application requires an internet connection to download the files needed from the system's Github repository.

1.2 Limitations

As of the moment, four strains are currently supported by the classifier. These are:

1. B.1.1.529 (Omicron)
2. B.1.617.2 (Delta)
3. B.1.621 (Mu)
4. C.37 (Lambda)

1.3 Installation

The following dependencies are required to begin the installation process:

1. Python 3.10.x (available at python.org with guide available at digitalocean.com) for installation of the other required packages via `pip`
2. Git (available at git-scm.com, with guides available at simplilearn.com and phoenixnap.com), for accessing the project files stored on the GitHub repository at https://github.com/bjorgkav/concreteml-covid-classifier.git
3. Windows Subsystem for Linux 2 (installation guide available at learn.microsoft.com) to allow support for Concrete-ML and the Dashing tool

1.3.1 Installation Process

1. If you are using WSL2, ensure that you have followed the "Existing WSL Install" section of Microsoft's guide to enabling Linux GUI support available at learn.microsoft.com.

   (a) Run the following commands on the WSL terminal to install X11, which is Linux's windowing system:

   ```
   sudo apt install x11-apps -y
   ```

   Listing 1: Command to install X11 and its apps

   (b) Next, run this command to set the $DISPLAY variable for your WSL environment:

   ```
   export DISPLAY=:0;
   ```

   Listing 2: Command to set the display variable

   (c) To verify if GUI apps now work on your WSL environment, run this command to open a calculator GUI application:

   ```
   xcalc
   ```

   Listing 3: Command to run a GUI calculator for testing GUI capabilities

2. After ensuring that Python and Git have been successfully installed and configured, clone the Git repository at https://github.com/bjorgkav/concreteml-covid-classifier.git using the following command:

   ```
   git clone https://github.com/bjorgkav/
   concreteml-covid-classifier.git
   ```

   Listing 4: Command to clone the project's Git repository

After cloning the Git repository, open the repository and ensure that `requirements.txt` is present.

3. Open your WSL terminal by typing "bash" in your file explorer's address bar. This will open WSL and automatically change its directory to your current directory.

4. Once it has opened, run the command:

```
pip install -r requirements.txt
```

Listing 5: Command to install the required packages

This will tell Python's package manager to install the specified versions of packages outlined in `requirements.txt`.

   (a) Ensure that you are using an Ethernet or Wi-Fi connection, or a sufficiently fast mobile network, as the package downloads may fail due to the connection timing out.

5. After installing the required packages in `requirements.txt`, the system should be ready to run. Other dependencies such as the Dashing tool binaries will automatically be downloaded by the client-side application once it is run.

2 Instructions for Use

2.1 Input File Structure

Clients will interact with our viral strain classification through the client-side GUI application, which takes one FASTA file as input for classification. Following the FASTA format defined by the National Center for Biotechnology Information at ncbi.nlm.nih.gov, the input file structure is:

1. The file begins with the FASTA definition line, which is included before the nucleotide sequence. It must begin with a carat (">") and should be followed by a unique sequence identifier (SeqID).

   - The FASTA definition line in this sequence should follow the following format:
     `>Reference/Database|AccessionID|DateCollected`
   - The SeqID should be unique for each nucleotide sequence.
   - The SeqID should not contain any spaces.
   - The SeqID should only contain the following characters: letters, digits, underscores (_), periods (.), asterisks (*), colons (:), hyphens (-), and number signs (#).

- All the information should be on a single line of text. The FASTA definition line should not include any hard returns.

2. The sequence begins after the FASTA definition line, and can contain returns. NCBI recommends that each line should be no more than 80 characters, and should only contain IUPAC symbols only, with "N" being used to symbolize ambiguous characters.

2.2 Classification Workflow

1. To begin classification, we must first ensure the server is running. In development, we do this by opening WSL in our file directory using "bash". Once the WSL terminal opens, we can then run the following command:

```
python3 manage.py runserver
```

Listing 6: Command to run the Django development server

Where `python3` is the alias for your Python installation. This command will run the project's Django development server. From there you should now be able to visit the server-side homepage (accessible by default at `http://127.0.0.1:8000/` or `localhost:8000`) to confirm that the server is running. This homepage provides general information about the application and its dependencies and supported strains, as well as providing links to its GitHub repository and dependencies.

2. On another WSL instance, use the `cd` command to change directories to the `client/GUI` directory. Run the client-side GUI application (`Client_GUI_App.py`). This can be done using the WSL terminal or using any IDE that you have installed. The WSL terminal command for running is:

```
python3 Client_GUI_App.py
```

Listing 7: Command to run client-side application from the terminal

This will run the client-side application, which will automatically download the required files before opening.

3. Click on the "Browse..." button in the GUI app to open a file dialog box. From here, you can select your FASTA input file (see 2.1 for details on formatting), and click on the "Submit for FHE Classification" button to preprocess, encrypt, and send your FASTA input file to the server for encrypted classification. It should then send it back to your client-side application, which will automatically decrypt and display your prediction results on the prediction output feed.

- The client-side application will also save your prediction results as a CSV file with your sequence's accession ID and classification date as its filename. This will allow you to view your results at a later time.