Supporting Information

High Fidelity Machine Learning-Assisted False Positive Discrimination in Loop-Mediated Isothermal Amplification Using Nanopore-Based Sizing and Counting

Ming Dong[1], Aneesh Kshirsagar[1], Anthony J. Politza[2] and Weihua Guan[1, 2*]

[1] Department of Electrical Engineering, Pennsylvania State University, University Park, Pennsylvania 16802, United States

[2] Department of Biomedical Engineering, Pennsylvania State University, University Park, Pennsylvania 16802, United States

* Correspondence should be addressed to wzg111@psu.edu

Contents

Supplementary Figures 2

Supplementary Tables 4

Supplementary Notes 5

Supplementary References 10

Supplementary Figures

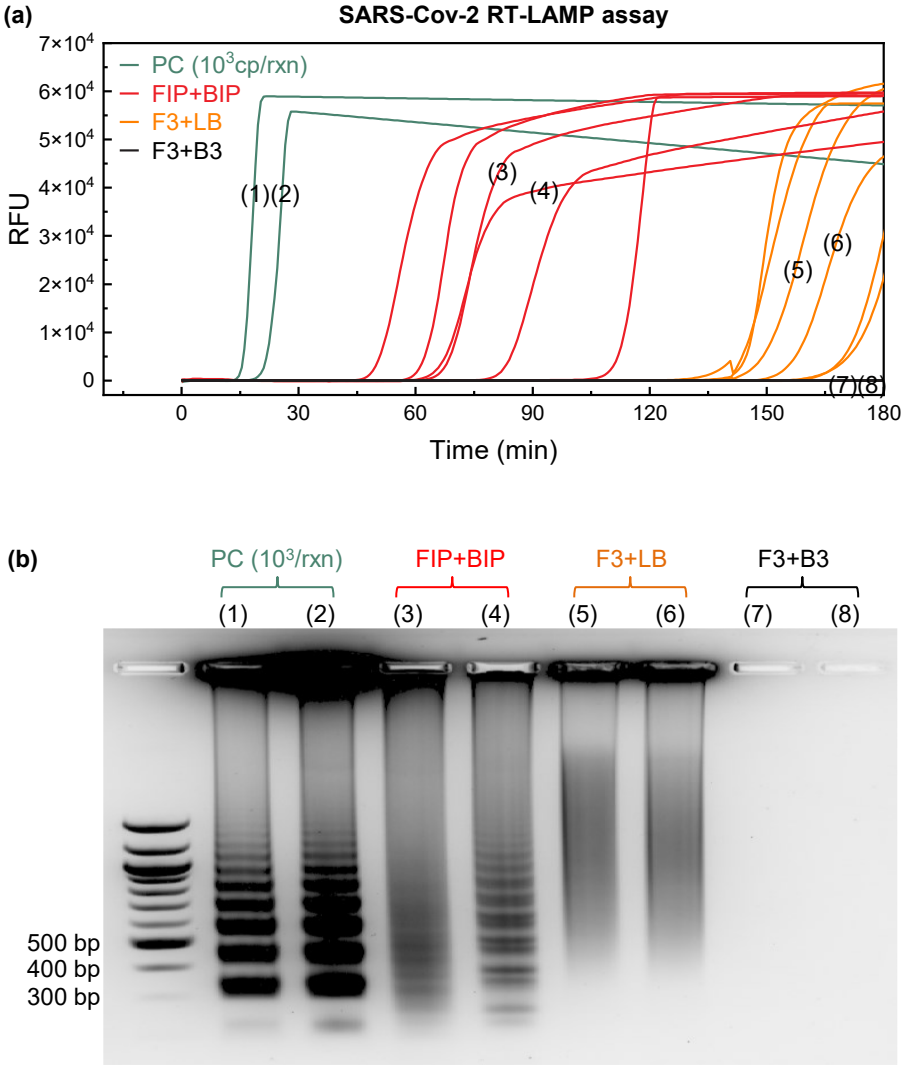
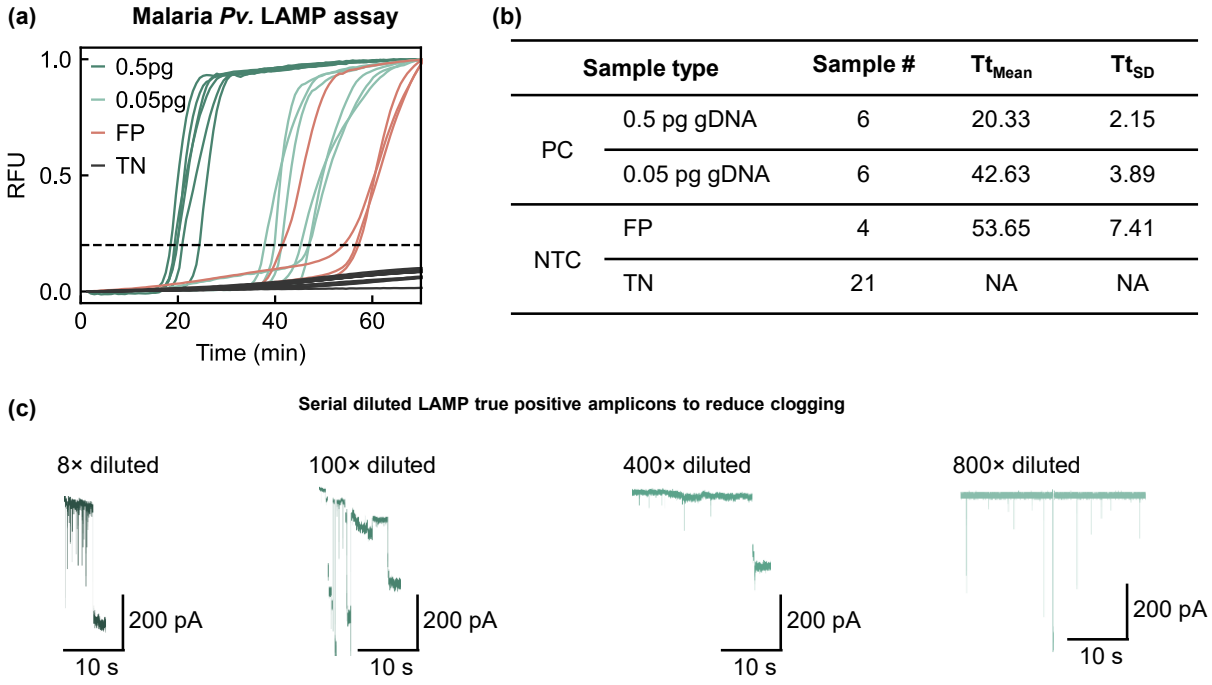


Figure S1. Analysis of primer-induced false positives in LAMP Assay. (a) Real-time RFU profiles from SARS-Cov-2 RT-LAMP assay using varied combinations of primers. Two positive controls (PC) include all six primers. Assays labeled "FIP+BIP" utilize only FIP and BIP primers without targets, as do "F3+LB" and "F3+B3". The primers-only tests exhibit a lag in amplification relative to the positive control, indicating reduced efficacy in primer-induced amplification. (b) Gel electrophoresis of resultant amplicons. The primer-induced amplicons present a distinct banding pattern in contrast to true positive amplicons.



Sample type		Sample #	Tt_{Mean}	Tt_{SD}
PC	0.5 pg gDNA	6	20.33	2.15
	0.05 pg gDNA	6	42.63	3.89
NTC	FP	4	53.65	7.41
	TN	21	NA	NA

Figure S2. Real-time RFU curves obtained from the Malaria LAMP assay, and the serial dilution of amplicons to mitigate nanopore clogging. (a) The normalized RFU curves of Malaria *Pv.* LAMP assays across varying target concentrations, with a set RFU threshold of 0.2. (b) The table provides assay statistics of positive controls (PC) and no template control (NTC). "FP" is false positives, "TN" is true negatives, and "Tt" is time-to-threshold. The overlapping Tt values between the 0.05 pg DNA samples and false positives underscore the difficulty of differentiating false positives using only fluorescence signals. All positive controls and false positive samples were used in nanopore sized counting experiment. (c) The serially diluted amplicons without adding calibrators were tested with the same nanopore device (I_{Base} = 7.5 nA at 400 mV bias), revealing prolonged unclogging times with increasing dilutions. A 500× dilution ensured a continuous 30 s unclogged measurement and sufficient event frequency in the nanopore device.

Supplementary Tables

Table S1. Primer sets for Malaria *P. vivax* LAMP assay.

Target	Primer	Sequences (5' to 3')
Malaria P. vivax[1]	F3	GGTACTGGATGGACTTTATAT
	B3	GGTAATGTTAATAATAGCATTACAG
	FIP	CCAGATACTAAAAGACCAACCCACCATTAAGTACATCACT
	BIP	GCTAGTATTATGTCTTCTTTCACTTAATATACCAAGTGTTAAACC
	LF	GATAACATCTACTGCAACAGG
	LB	CTACTGTAATGCATCTAAGATC

Table S2. Primer sets for SARS-Cov-2 RT-LAMP assay.

Target	Primer	Sequences (5' to 3')
SARS-Cov-2[2]	F3	AACACAAGCTTTCGGCAG
	B3	GAAATTTGGATCTTTGTCATCC
	FIP	TGCGGCCAATGTTTGTAATCAGCCAAGGAAATTTTGGGGAC
	BIP	CGCATTGGCATGGAAGTCACTTTGATGGCACCTGTGTAG
	LF	TTCCTTGTCTGATTAGTTC
	LB	ACCTTCGGGAACGTGGTT

Table S3. Primer sets for respiratory syncytial virus type A (RSV A) RT-LAMP assay.

Target	Primer	Sequences (5' to 3')
RSV A2[3]	F3	GAGTTGAAGGGATTTTTGCA
	B3	TGGGTTGTTCAATATATGGTAGA
	FIP	TAACTGATTTTGCTAAGACCCCCGGATTGTTTATGAATGCCTATGG
	BIP	CAAGCAGAAATGGAACAAGTTGTGCTGCTTCTCCACCCAATT
	LF	CACCGTAACATCACTTG
	LB	GAGGTGTATGAGTATGCTCAGA

Supplementary Notes

Note S1. Assay Information

LAMP assay. The total volume of the LAMP assays contains a 24 µl master mix and 1 µl Malaria *Pv.* gDNA. The master mix includes isothermal buffer, nuclease-free H_2O, $MgSO_4$ (7 mM), SYTO-9 green (0.5 µM), deoxyribonucleotide triphosphates (dNTPs, 1.4 mM), Bst 2.0 DNA polymerase (0.4 U/µl), primer sets (0.2 mM F3 and B3c, 1.6 mM FIP and BIP, 0.8 mM LF and LB). The reaction was performed at a constant temperature of 65 ºC using a benchtop PCR instrument (Bio-Rad CFX96).

RT-LAMP assay. The total volume of the RT-LAMP assays contains a 24 µl master mix and 1 µl RNA sample. The master mix includes isothermal buffer, nuclease-free H_2O, $MgSO_4$ (7 mM), SYTO-9 green (0.5 µM), deoxyribonucleotide triphosphates (dNTPs, 1.4 mM), Bst 2.0 DNA polymerase (0.4 U/µl), Warmstart reverse transcriptase (0.3 U/µl), primer sets (0.2 mM F3 and B3c, 1.6 mM FIP and BIP, 0.8 mM LF and LB). The reaction was performed at a constant temperature of 65 ºC using a benchtop PCR instrument (Bio-Rad CFX96).

Note S2. Kinetic Model for LAMP

Utilizing the target-induced specific amplification pathway inherent to LAMP assays as a foundation, our study developed a simplified kinetic model to explore the effects of varying parameters on amplicon profiles. This model operates under the assumptions of constant polymerase activity and an inexhaustible supply of reaction reagents. We employed MATLAB to simulate and record the amplification process of DNAs over time. The core data structure, "AmpliconInfo", captures the attributes of amplicons: their number, the time of their last modification, and their length. The entire amplification process lasts 50 seconds. The code checks existing amplicons for each time unit and determines whether they need to be updated (for mother amplicons) or if a new daughter amplicon is generated. Daughter amplicons are produced by specific (FIP/BIP/LF/LB) primers, and their lengths are incremented by one unit. The process iterates for a number of branches, and each branch is initiated by a newly replicated DNA target every t_R. After evaluating all amplicons for a given branch, the data is replicated based on the branch number to account for the exponential growth of the amplicons. The results for each branch are then saved to individual CSV files in a specified directory. Then, a histogram

will be generated to visualize the distribution of amplicon lengths.

MATLAB code for the LAMP model:

```
clc; clear all;
AmpliconInfo = []; % [number time length]
Time = 50; %total amplification time (s)
t_R = 2; %self-replication time for mother amplicon (s)
L = 100; %Unit spacing length (bp)
t_E = L/50; %elongate time for one spacing (s), extending 50 bp per second
branch = floor(Time/(1* t_R))-1;%total number of branches
% the preparation path will generate new branch every t0
outputpathname ='C:\Users\desired path';

for br = 0:1:branch
    T=Time-br*1* t_E; %T is the each branch amplification time

    i=1; %T is the each branch amplification time
    t=0;
    AmpliconInfo(1,1)=1;
    AmpliconInfo(1,2)=0;
    AmpliconInfo(1,3)=1;
    newi=1;
    while t<=(T- t_E) % at each t, check all existing amplicons
        j=1; % starting from the first amplicons
        t=t+ t_E;
        while j<=i
            if t-AmpliconInfo(j,2)==(3*AmpliconInfo(j,3)+1)* t_E %Mother amplicon
                AmpliconInfo(j,1)=j;
                AmpliconInfo(j,2)=t;
                AmpliconInfo(j,3)=AmpliconInfo(j,3); %length will not change
            end
            if t-AmpliconInfo(j,2)==(2*AmpliconInfo(j,3)+1)* t_E %Daughter amplicon
                AmpliconInfo(newi+1,1)=newi+1; %Register as a new amplicon
                AmpliconInfo(newi+1,2)=t;
                AmpliconInfo(newi+1,3)=AmpliconInfo(j,3)+1; %Elongation unit spacing
                newi=newi+1;
            end
            j=j+1;
        end
        i=newi;
    end
    AmpliconInfoTemp=[];
    for daughter=0:1:(2^br-1)
        AmpliconInfoTemp=[AmpliconInfoTemp;AmpliconInfo];
    end
    AmpliconInfo=AmpliconInfoTemp;
    %%%%Save Data as CSV%%%%
    sample=strcat('branch',string(br),'.csv');
    outputpath=fullfile(outputpathname,sample);
    AmpliconInfo(:,2)=AmpliconInfo(:,2)+br*t1;
    csvwrite(outputpath, AmpliconInfo);
    AmpliconInfo=[];
end
```

```
    %%%%Compose Data and Plot Histogram%%%%
allFiles                    =           dir(outputpathname);
numFiles                    =           size(allFiles,1);
allData                     =           [];
for i = 1:numFiles
    thisFileName            =           allFiles(i).name;
    if ~allFiles(i).isdir  &&  allFiles(i).bytes>0
        thisFileData        =           csvread([PathName '\' thisFileName]);
        allData             =           [allData; thisFileData];
    end
end
allData                                 =           sortrows(allData,1);
Allampliconinfo                         =           allData(:,3);
figure;
histogram(Allampliconinfo)
xlabel('Length (# of spacing)');
ylabel('Counts');
```

Note S3. Machine Learning-Assisted Classification

The machine learning method processes and analyzes data from two distinct categories: 'true positives' and 'false positives' from LAMP amplicon profiles. These categories are represented by datasets stored in separate folders containing multiple Excel files. The first step in the process is feature extraction, where specific characteristics of the data are computed. The features include mean, standard deviation, skewness, and kurtosis of frequency values. Notably, the script ensures that all frequency values are non-negative by replacing negative values with zero, which is crucial for accurate feature calculation. Additionally, it will identify peaks in the histogram of the amplicon, providing insights into the data distribution. Once features are extracted, the script transitions into a machine learning phase, employing a support vector classifier (SVC) with a linear kernel. The Leave-One-Out cross-validation method, a rigorous testing approach, is used to evaluate the model, which is especially beneficial for small datasets. This method iterates through the dataset, using each point once as a test set while training the model on the rest. The model's performance could be assessed using the receiver operating characteristic (ROC) curve and the area under the curve (AUC). The script also calculates the model's accuracy to evaluate its overall performance.

Python code for the machine learning-assisted classification

```
import pandas as pd
import numpy as np
import glob
from sklearn.model_selection import LeaveOneOut
```

```
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.svm import SVC
from scipy.stats import skew, kurtosis
from scipy.signal import find_peaks
import matplotlib.pyplot as plt

# Function to extract relevant features
def extract_features(df):
    df['frequency'] = df['frequency'].clip(lower=0)
    # Replace all negative values in the 'frequency' column with 0 to avoid
    misinterpretation of data and inaccurate features extraction.
    features = {
        'mean': np.mean(df['frequency']),
        'SD': np.std(df['frequency']),
        'skewness': skew(df['frequency']),
        'kurtosis': kurtosis(df['frequency']),
    }

    # Finding peaks
    hist_data, bin_edges = np.histogram(df['amplicon_size'], weights=df['frequency'],
bins=160)
    peaks, _ = find_peaks(hist_data)
    features['num_peaks'] = len(peaks)  # Adding number of peaks as a feature

    if len(peaks) > 0:
        peak_intensities = hist_data[peaks]

        # Calculating and adding relative peak intensities as features
        max_peak_intensity = np.max(peak_intensities)
        features['max_peak_intensity'] = max_peak_intensity
        features['mean_peak_intensity'] = np.mean(peak_intensities)
        # Adding peak relative intensity
        features['mean_peak_relative_intensity'] = max_peak_intensities /
        mean_peak_intensity
    else:
        # Handle the case where no peaks are found
        features['mean_peak_relative_intensity'] = 0

    return pd.Series(features)

# Data Loading and Feature Extraction
data = []
file_name_dict = {'true_positive': [], 'false_positive': []}  # Dictionary to keep
track of file paths for each label

for label, folder in [('true_positive', r'C:\Training_Source\48_Samples\True
positives'),
                      ('false_positive', r'C:\Training_Source\48_Samples\False
positives')]:
    for file_path in glob.glob(f"{folder}\\*.xlsx"):
       df = pd.read_excel(file_path, usecols=[0, 1], header=None,
       names=['amplicon_size', 'frequency'])
```

```
        # Displaying the loaded Excel data (first 5 rows)
        print(f"Data from: {file_path}")
        print(df.head(1))
        print("\n" + "="*40 + "\n")

        features = extract_features(df)
        features['label'] = label
        data.append(features)

data_df = pd.DataFrame(data)

# # Separate DataFrames based on label
true_positives_df = data_df[data_df['label'] == 'true_positive']
false_positives_df = data_df[data_df['label'] == 'false_positive']

# # Output features to Excel files for parallel coordinate plots
true_positives_df.to_excel(r'C:\true_positives_features.xlsx', index=False)
false_positives_df.to_excel(r'C:\false_positives_features.xlsx', index=False)
X = data_df.drop(columns=['label'])
y = (data_df['label'] == 'true_positive').astype(int)
model = SVC(kernel='linear', probability=True)
loo = LeaveOneOut()
predictions, truths = [], []

for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    model.fit(X_train, y_train)
    probas_ = model.predict_proba(X_test)
    predictions.append(probas_[:, 1])
    truths.append(y_test.iloc[0])

# ROC curve
fpr, tpr, thresholds = roc_curve(truths, predictions)
roc_auc = auc(fpr, tpr)
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.4f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()

accuracy = accuracy_score(truths, [1 if p >= 0.5 else 0 for p in predictions])
# Note that we threshold predictions here at 0.5 for computing accuracy
print(f"LOO Accuracy: {accuracy:.4f}")
model.fit(X, y)
```

Supplementary References

(1) Tang, Z.; Choi, G.; Nouri, R.; Guan, W. Loop-Mediated Isothermal Amplification-Coupled Glass Nanopore Counting Toward Sensitive and Specific Nucleic Acid Testing. *Nano Lett.* **2019**, *19*, 7927–7934.

(2) Tang, Z.; Nouri, R.; Dong, M.; Yang, J.; Greene, W.; Zhu, Y.; Yon, M.; Nair, M. S.; Kuchipudi, S. V.; Guan, W. Rapid Detection of Novel Coronavirus SARS-CoV-2 by RT-LAMP Coupled Solid-State Nanopores. *Biosens. Bioelectron.* **2022**, *197*, 113759.

(3) Takayama, I.; Nakauchi, M.; Takahashi, H.; Oba, K.; Semba, S.; Kaida, A.; Kubo, H.; Saito, S.; Nagata, S.; Odagiri, T.; Kageyama, T. Development of Real-Time Fluorescent Reverse Transcription Loop-Mediated Isothermal Amplification Assay with Quenching Primer for Influenza Virus and Respiratory Syncytial Virus. *J. Virol. Methods* **2019**, *267*, 53–58.