FLIM Playground: An interactive, end-to-end graphical user interface for analyzing single-cell fluorescence lifetime data

Wenxuan Zhao[1], Kayvan Samimi[1], Melissa C. Skala[1,2], Rupsa Datta[1*]

[1]Morgridge Institute for Research, Madison, WI, USA

[2]Department of Biomedical Engineering, University of Wisconsin, Madison, WI, USA

*correspondence: rdatta@morgridge.org

ABSTRACT

Fluorescence lifetime imaging microscopy (FLIM) is sensitive to molecular environments and enables high-resolution mapping of cellular heterogeneity. Yet, the analytical journey from raw photon decays to biological insight remains fragmented by multi-step data extraction and siloed downstream analyses that introduce friction and propagate errors. We present FLIM Playground, the first interactive graphical platform that unifies single-cell FLIM workflows, embeds user checks at each stage, and offers diverse options for user preferences. Built in Python and available open-source, FLIM Playground runs on all major operating systems as a ready-to-run app and can also be deployed on the web. The Data Extraction section interactively collects and checks field-of-view metadata, calibrates either via instrument response function (IRF) shift or reference dye, and extracts single-cell fluorescence lifetime features, both fitted and phasor, along with morphology and texture features across all channels. Multiple datasets spanning time, samples, or treatments can be merged through an interface that assigns categorical labels for downstream analyses. The Data Analysis section accepts either outputs from the Data Extraction or user-provided datasets. It provides visual analytic methods that share common interactive controls to visualize and filter large datasets in real time through a graphical user interface (GUI). Validation of lifetime fitting with FLIM Playground was performed by comparison with SPCImage using two-photon autofluorescence FLIM of Panc1 cells. Additionally, two-photon autofluorescence FLIM of Panc1 and MCF7 cells treated with metabolic inhibitors were used to demonstrate FLIM Playground features. By adopting best practices and offering flexibilities through interactivity, FLIM Playground accelerates hypothesis-driven discovery and promotes reproducibility.

1. INTRODUCTION

Fluorescence lifetime imaging microscopy (FLIM) is sensitive to changes in fluorophore microenvironment including conformational changes with protein binding and the presence of quenchers. Coupled with modern automated cell-segmentation methods[1–4], FLIM enables single-cell analyses that reveal biological heterogeneity.

A typical FLIM experiment spans multiple imaging sessions, each containing multiple multi-channel fields-of-view (FOVs). Each acquisition channel contains multiple cell-level regions of interest (ROIs), defined by ROI masks and composed of pixel sets. At the pixel level, fluorescence decays can be fitted[5,6] or transformed into phasor features[7–9]. ROI masks or pre-aggregated cell level decays[10] enable single-cell resolution mapping of heterogeneity through fitting[11], phasor[12], morphology and texture[13] extraction. The channel level introduces fluorophore-specific inputs, enabling per-channel extraction of

single-cell numerical features. The field-of-view level aggregates information across channels, allowing unique single-cell identifiers to be linked with their imaging context. Finally, the experiment level takes qualitative category labels of FOVs and incorporates them into downstream analysis methods to gain biological insights. Together, pixels, cell ROIs, channels, fields of view, and experiments constitute the hierarchical data levels in FLIM analysis, each representing a distinct unit of information with progressively richer biological semantics.

While a diverse set of tools—both open-source and commercial, ranging from libraries to code-free graphical user interfaces—offer alternative methods and flexibility for FLIM analysis, they typically address only subsets of the data levels. As a result, users often write custom code to reformat inputs, process outputs, and explore alternative methods, leading to a fragmented landscape of FLIM analysis.

FLIM Playground aims to provide a unified framework for integrating diverse methods and input types across data levels, accessible through a cross-platform, interactive, and code-free graphical user interface (GUI). It spans the full pipeline: from field-of-view metadata organization, through calibration with instrument response functions (IRF) or reference dyes, to extraction of single-cell identifiers, numerical features (lifetime fitting and phasor, morphology, and texture) and categorical features, followed by visualization and statistical modeling. Embedded validation checks guide users at every step, while interactive widgets and a built-in repertoire of analytical methods encourage hypothesis-driven and iterative exploration of large datasets. Its modular architecture enables incorporation of new methods and algorithms as they evolve, and, being open-source and Python-based, facilitates adoption and extension by other labs and leverages the rich Python package ecosystem.

To ensure reliability, we compared lifetime fitting with FLIM Playground against the widely used commercial SPCImage package[6] using two-photon autofluorescence FLIM of cells. We further illustrate the capabilities of FLIM Playground on autofluorescence FLIM of cancer cell lines (Panc1 and MCF7) treated with metabolic inhibitors.

2. METHODS

FLIM Playground is organized into two main sections: Data Extraction and Data Analysis. Data Extraction derives single-cell features from raw FLIM images, while Data Analysis can either process these outputs to complete the workflow or operate directly on single-cell datasets extracted by user's own workflow.

Both sections focus on the three feature classes: identifiers (cell and field-of-view IDs), categorical features (discrete labels to cell groups), and numerical features (quantitative measures that capture differences or similarities between groups) (Table 1). Each of the three Data Extraction stages targets one feature class and provides interactive interfaces with validation checks and status feedback. Data Analysis builds a shared library of interactive interface components that operate across feature classes, while method-specific components further support a variety of statistical modeling approaches. Both sections are designed to support the application of alternative methods to diverse input types, with the flexibility to be readily extended.

Data Levels	Data Processing	Feature Classes
Experiment	Assign category labels (e.g., treatment, cell line, etc.) to each cell; perform data analysis on labeled cell groups.	Categorical features
Field of View	Assign FOV identifier to each cell, obtain unqiue cell identifier	Identifiers
Acquisition Channel	Apply **per-channel** ROI mask, calibration and extraction methods to cell-level signals	Numerical features
Cell ROI	Aggreate pixel-level decays to obtain cell-level signals	**Single-cell resolution**
Pixel	Read pixel-level decays encoded in different formats	N/A

Increasing complexity

Table 1. FLIM Playground processes at each data level to get single-cell features. Each data level in the hierarchy is composed of a set of instances from the level below (e.g., a cell is composed of a set of pixels). FLIM Playground manages this complexity by reading pixel-level decays encoded in vendor-specific formats (e.g., Becker & Heckel), and aggregate the decays that belong to the same cell ROI to obtain the cell-level signals (i.e., cell-level decay and cell intensity image). In this way, we can get single-cell resolution. At each acquisition channel, FLIM Playground flexibly handles the per-channel masks, calibration procedures, and methods (e.g., fitting, phasor, texture, and morphology) to extract numerical features (yellow). FLIM Playground organizes metadata for FOVs and assigns to cells the FOV they come from and their unique identifiers (red). Category labels (blue) such as treatment and time points are biologically meaningful and FLIM Playground can assist user in assigning those to each cell and use them to perform data analysis.

FLIM Playground employs the interactive graphic user interface (GUI) library Streamlit[14], which supports both local and web deployment. PyInstaller[15] bundles the software into executables that run locally on major operating systems (ready to use executables for Windows and Mac are available in the FLIM Playground Github[16] repository). A detailed online manual[17] is also available for user reference.

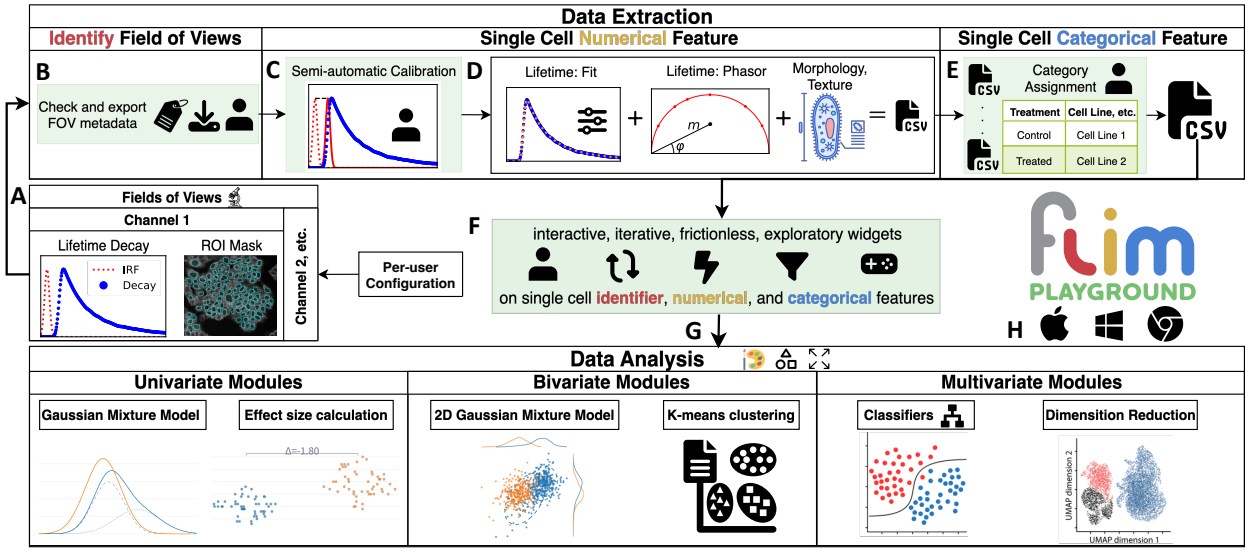

Fig 1. System overview of FLIM Playground. After **(A)** loading a saved configuration, **(B) Data Extraction** collects and validates channel inputs across all fields of views, **(C)** calibrates for IRF shifts or uses reference standards, and **(D)** extracts single-cell lifetime (fitting and phasor), morphology, and

texture features in one step. **(E)** To bridge the data extraction with downstream analysis, FLIM Playground combines multiple single-cell datasets and assists users with assigning category labels to each cell. **(F)** The three feature classes include identifiers (red), numerical features (yellow), and categorical features (blue), are transformed into interactive widgets shared by all analysis methods. **(G) Data Analysis** includes univariate, bivariate, and multivariate modules. **(H)** FLIM Playground can either be deployed online in major operating systems or as a data analysis web application. Interactive interfaces are indicated by green backgrounds.

2.1 The Data Extraction section extracts single-cell features using the modularized framework

A channel-centric framework provides flexibility for user to assign per-channel masks that might focus different cell ROI, calibration procedures, and extraction methods to extract numerical features (Table 1). These assignments are managed through a configuration interface (Supp Fig 1), so the configuration can be set up once and reused across future analyses.

2.1.1 Configuration Interface

Based on the number of channels specified by the user, the interface renders a column for settings within each channel. Each channel can be assigned a name (e.g., fluorophore label), an imaging modality, and a list of feature extractors.

Two imaging modalities are supported. In *FLIM mode*, the signal is time-resolved and corresponds to one of the decay types described below. In *intensity-only mode*, the signal is a spatial intensity map without a lifetime axis (e.g., TIFF/TIF format), enabling mixed-modality analyses within the same field of view and supporting analysis of non-FLIM images.

Each acquisition channel can be associated with a subset of extractors rather than a fixed choice. Lifetime-based feature extractors, fitting or phasor, can be applied to FLIM channels, while morphology- and texture-based extractors are available for all imaging modalities. These extractors will be introduced in detail in later sections. Based on the selected imaging modality and feature extractors, the interface prompts the user to specify file suffixes for each required input file type (e.g., raw decay, ROI mask, calibration file), which are later used during the FOV identification stage. Channels can either share a common ROI mask or use channel-specific masks to target distinct compartments within a cell.

Beyond channel-specific settings, several parameters apply across all channels. Three decay types are supported: (1) 3D/4D decay, consisting of spatial dimensions and the fluorescence decay axis, with an optional channel dimension capturing multiple fluorophores (time-lapse acquisitions are not yet supported). These image data are stored in vendor-specific formats, and FLIM Playground supports those from Becker & Hickl and PicoQuant. (2) 2D decay, where each row of a tabular sheet corresponds to a single cell and each column to a time bin of its decay, as in fluorescence lifetime flow cytometry[10]; and (3) 3D/4D pixel-prefit decay, representing pixel-level lifetime features pre-fit by open-source or commercial software such as SPCImage. Two calibration methods are available for phasor analysis: either by shifting the IRF or by using a reference dye with

known lifetime[18]. Users may also specify a list of categorical features for extraction, which will later be recognized in the Data Analysis section. A few more settings are introduced in the manual[17], Ch. 3.

The next steps after specifying the reusable configuration are the three stages of Data Extraction, each focusing on extracting one feature class (Fig 2).

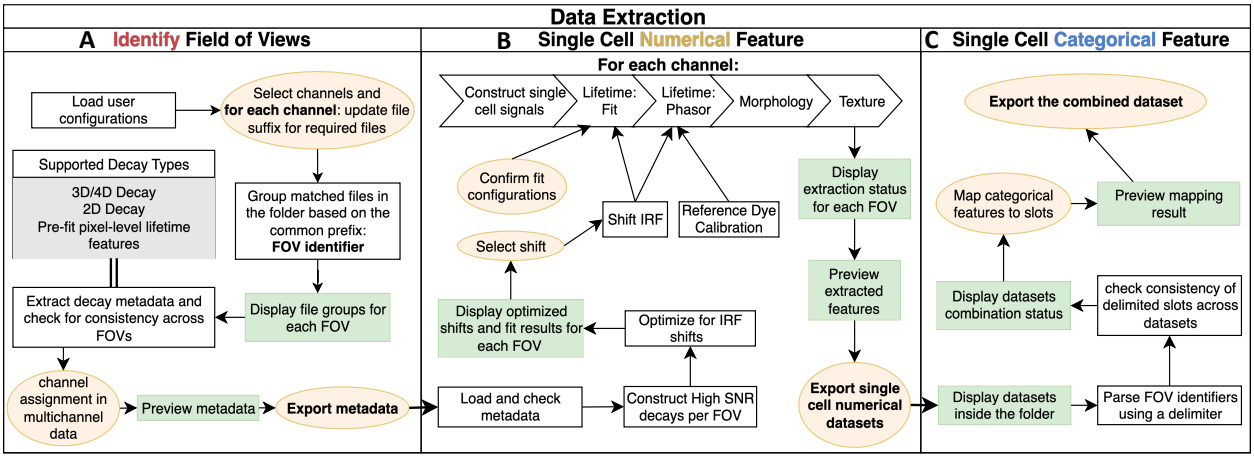

Fig 2. A high-level overview of the Data Extraction workflow. This section is divided into three stages **(A, B, C)**. A short description of steps involved in each stage is shown here, with detailed descriptions provided in following sections. The white boxes are internal processing steps not shown to the user. The yellow oval-shaped boxes are interactive controls user can change. Green boxes are interfaces that provide feedback to the user.

2.2 Identify Fields of View

Users are prompted to enter the file path to the folder that contains all files necessary for the data extraction (Fig 2A, Supp Fig 2). Per-channel and cross-channel settings are loaded from the configuration, and most can be updated at this stage, including the file suffixes for each required input file type of each channel. FLIM Playground recursively searches the folder path for files that end with the first file suffix of the first channel. The prefixes of the matched files are defined as FOV identifiers (i.e., file name minus the first suffix), under the assumption that **all files associated with an FOV share the same prefix.** All other files across channels are then located by appending each suffix to the FOV identifier. The only exception is calibration files, which are not FOV-specific and are therefore searched based solely on their file suffixes. For each FOV, the search status is reported. A status of *Success* indicates that all required files were located, and the FOV is then recorded into a metadata file together with the paths to its files. If a required file type is missing, its type and expected filename are displayed. A status of *Duplicate* is shown when multiple files with the same name are detected.

FLIM Playground then reads the metadata of all decay files. The duration of time windows between laser pulses and the number of time bins per pulse interval are cross-checked across all channels for consistency. The time bins are also validated against the lifetime axis of the calibration files. In addition, the spatial dimensions of the decay files are cross-checked with the ROI masks to ensure consistency. For each acquisition channel, if the corresponding decay file contains more than one non-empty channel, the user is prompted to select the appropriate channel number. The sdtfile[19] and ptufile[20] Python

libraries are used to read decay data from Becker & Heckel and PicoQuant files. Channel numbers, FOV spatial dimensions, duration, and time bins are recorded into the metadata file. The imaging modality and feature extractors for each channel are also recorded. A preview of the metadata file is displayed, which can be exported in comma-separated values (CSV) format. The file is simultaneously cached in the system to enable seamless progression to the next stage.

2.3 Extract Single-Cell Numerical Features

Using either the cached metadata file from the previous step or a newly uploaded file, FLIM Playground initiates numerical feature extraction as a two-step process: calibration followed by feature extraction (Fig 2B). Calibration is required whenever lifetime fit or phasor analysis is selected, since at acquisition raw fluorescence decays are convolved with the IRF. Calibration is carried out separately for each acquisition channel.

2.3.1 Fit Calibration

In *fit calibration*, IRF time shifts are estimated through reconvolution fitting, where the shift is treated as a free parameter optimized jointly with lifetimes and amplitudes using high signal-to-noise (SNR) decay curves from each FOV (Supp Fig 3). For each FOV, high SNR decays are constructed by summing the decay curves of all non-background pixels defined by the ROI mask. The brightest cells are selected as high SNR decays if the decay type is 2D decay. Reconvolution fitting models the measured fluorescence decay as a sum of exponential functions—reflecting the characteristic lifetimes of fluorophores—and fits this model after convolution with the instrument response function (IRF) to account for temporal broadening introduced during acquisition.

The time axis ($\mathbf{t}$) of the exponential functions is defined by sampling steps (Δt), which is the ratio between time window duration (T) and the number of time bins (N) (Eq. 1).

$$\mathbf{t} = [0,\ \Delta t,\ 2\Delta t,\ \dots,\ (N-1)\Delta t], \qquad \Delta t = T/N \qquad (1)$$

A linear convolution ($*$) is performed between the multi-exponential model (A_i are amplitudes, τ_i are lifetimes of each component) and the shifted IRF ($\mathbf{IRF}_s$) with truncation ($0{:}N$) (Eq. 2). The offset parameter (b) is also optimized. Together, this yields the fitted curve ($\hat{\mathbf{y}}$). FLIM Playground supports fitting models with up to three *components*.

$$\hat{\mathbf{y}} = \left(\sum_{i=1}^{n} A_i\ e^{-\mathbf{t}/\tau_i} * \mathbf{IRF}_s\right)_{0:N} + b \qquad (2)$$

Parameter optimization is performed by minimizing the difference between the fitted curve ($\hat{\mathbf{y}}$) and the measured curve ($\mathbf{y}$), quantified by either Maximum Likelihood Estimation (MLE) or Least Square (LS) as a *cost metric*[6]. The fitting procedure also allows users to specify *time gates* $[t_s, t_e]$, restricting the optimization to a selected temporal window of the decay. Maximizing MLE is mathematically equivalent to minimizing negative log-likelihood function (NLL) (Eq. 3). θ denotes all parameters to be optimized; $\mathbf{1}_{[t_s,t_e]}(t_k)$ is the indicator function that is 1 if a time bin is within the time gates and 0 otherwise.

$$\mathrm{NLL}(\theta; t_s, t_e) = -\sum_{k=0}^{N-1} \mathbf{1}_{[t_s,t_e]}(t_k)\,[y_k \log\widehat{y_k} - \widehat{y_k}] \qquad (3)$$

LS captures the sum of the squared differences between the measured curve and the decay curve within the time gates (Eq. 4).

$$\mathrm{LS}(\theta; t_s, t_e) = \sum_{k=0}^{N-1} \mathbf{1}_{[t_s,t_e]}(t_k)\,(y_k - \widehat{y_k})^2 \qquad (4)$$

One of the two cost metrics (MLE or LS), chosen by the user, is fed into lmfit[21], a Python generic curve fitting library to be minimized. FLIM Playground provides three *optimization modes* that balance speed against robustness to local minima. Global optimization employs the differential evolution algorithm[22], a derivative-free, population-based but relatively slow global optimizer. Local optimization applies the Levenberg–Marquardt[23] algorithm when the chosen metric is least square, or the Nelder–Mead[24] algorithm otherwise. Hybrid optimization, the most time-consuming, combines both approaches by first using differential evolution to generate a reliable initial guess, followed by local refinement.

For each high-SNR decay curve, a shift value is optimized and displayed as a point in an interactive scatter plot. By clicking on a point, the user can inspect the measured decay alongside the fitted curve, together with the fitting metric and estimated parameters. This interface allows users to evaluate the fits, select a fixed shift to apply across all FOVs, or adjust the time gates to trigger re-optimization. Once the user confirms the *number of components* of the exponential model, *cost metric*, *fitting mode*, *time gates*, and shifts for all channels, these settings are stored in the metadata file and applied to extracting fitting features.

2.3.2 Phasor Calibration

When calibration is performed using the instrument response function (IRF), the temporal shift for each high-SNR decay is estimated by maximizing the cross-correlation between the IRF and the corresponding decay. Similar to fit-based calibration, all estimated shifts are plotted, and the user can select a fixed shift to apply across all decays. In addition to shifting the IRF by the chosen value, each decay curve is corrected by subtracting an offset—defined as the mean of the last tenth percentile of the decay tail—and clipped to zero if negative values occur after subtraction. The raw phasors of the decay are divided by IRF using complex division to obtain the calibrated phasor.

Because the reference dye was measured in the same system as the decay curves and its lifetime is known, neither IRF shift nor offset correction is required. In this case, phasor calibration is performed using the *lifetime.phasor_calibrate* function from the PhasorPy[9] library.

2.3.3 Feature Extraction

For each channel, the selected feature extractors specified in the previous stage are applied to each cell ROI (Supp Fig 4). A single-cell decay curve summed from all the decays of its pixels[11] is fitted with an exponential model and/or transformed into phasor coordinates.

The *Lifetime Fit* extractor applies reconvolution fitting to the decay curve of each cell using the shifted IRF, with fit settings specified by the user in the calibration step. In addition to saving the absolute amplitudes (A_i), it normalizes these amplitudes into fractions (α_i) (Eq. 5) and computes the mean lifetime (τ_m) (Eq. 6).

$$\alpha_i = \frac{A_i}{\sum_{j=1}^{n} A_j} \qquad (5)$$

$$\tau_m = \sum_{i=1}^{n} \alpha_i \, \tau_i \qquad (6)$$

It can also compute cell-level lifetime fitting features when pre-fit pixel-level results are available, by averaging the pixel-level features within each cell ROI.

The *Lifetime Phasor* extractor provides a fast, model-free view of lifetimes by transforming fluorescence decays into phasor coordinates through a Fourier transform. For each cell, raw phasors are computed from the fluorescence decay for the first and second harmonics, and they are calibrated using the procedures described in *Phasor Calibration* to obtain calibrated phasor coordinates (g, s). Phasor-derived lifetime parameters are also computed, including phase (ϕ), modulation (m), tau phase (τ_ϕ), and tau modulation (τ_M)[7,25,26].

The *Morphology* extractor computes single-cell morphological features of the cell ROI mask, including area, perimeter, solidity, eccentricity, major axis length, minor axis length, and circularity. The list can be readily extended.

The *Texture* extractor uses the single-cell intensity image to extract three classes of texture features that can be easily extended. For example, $granularity_n$ is the percentage of intensity removed when bright objects of n pixels in diameter are removed. A disk of radius n is used to perform a morphological opening on the cell intensity image, and the difference between the original image and the opened image corresponds to the removed bright objects. Five granularity-based features are calculated: $n = 1,3,5,7,9$. *Mass displacement* calculates the displacement, as Euclidean distance in pixels, of the intensity-weighted cell ROI centroid from the geometric centroid of the cell ROI. To calculate *radial distribution*, each cell intensity image is partitioned into four concentric rings, and the intensity fraction over the total intensity for each ring is calculated.

Each cell is assigned a unique identifier by concatenating its field-of-view (FOV) identifier with the corresponding label from the ROI mask, and this identifier is used as the row index. Each row then aggregates all features extracted across channels and feature extractors. The extraction status for each FOV is tracked with a progress bar, and the results are displayed for preview before being exported.

2.4 Extract Single-Cell Categorical Features

In this stage FLIM Playground merges cell-level datasets produced by the numerical feature extraction stage into a unified table and assigns categorical features to each cell (Fig 2C, Supp Fig 5). FLIM Playground searches the user-designated folder for CSV files, verifies the uniqueness of cell identifiers. Datasets are then merged by aligning shared

columns and concatenating rows. FOV identifiers are assumed to consist of segments delimited in a consistent manner, where each segment or combination of segments encodes a categorical feature (e.g., treatment, time point). Under this assumption, FLIM Playground parses the segments and presents them to the user, who interactively assigns them to categorical features with a live preview of the results. The resulting merged dataset, enriched with categorical features, can be exported for downstream analysis.

2.5 Data Analysis provides an interactive interface for diverse analysis methods

After single-cell datasets are extracted—either through Data Extraction or by the user's own methods—the Data Analysis section provides a suite of analysis methods (**Table 2**) to help the user gain biological insights. To ensure users can seamlessly apply alternative analysis methods across datasets with varying features, Data Analysis offers a shared interface built on a library of interactive widgets, allowing new modules to be easily added.

Analysis class & Numerical feature selection	**Modules**	**Filter by**	**Color by**	**Point-based**	**Built-in modeling methods**
Univariate — Select *one* feature from a feature group.	Feature Comparison	✓	✓	✓	Effect size calculation (Glass' Delta & Cohen's d)
	Feature Histogram	✓	✓	✗	Gaussian Mixture Model & Heterogeneity-index
Bivariate — Two univariate selectors.	2D Feature Distribution	✓	✓	✓	2D Gaussian Mixture Model & Linear Regression
Choose a channel and a harmonic to select phasor coordinates.	Phasor Analysis	✓	✓	✓	K-means Clustering
Multivariate — Select multiple features from each feature group.	Dimension Reduction	✓	✓	✓	PCA, t-SNE, UMAP with user-adjustable hyperparameters
	Classification	✓	✓*	✗	Random Forest, SVM, Logistic Regression, Gradient Boosting

Table 2. Modules offered in FLIM Playground Data Analysis. Modules are grouped into three analysis classes—Univariate, Bivariate, and Multivariate—based on the number of numerical features required, with custom feature-selection widgets implemented for each class (yellow). Each module pairs an interactive visualization with a statistical modeling method. *Filter by* and *Color by* are interactive widgets derived from categorical features (blue) and are shared by all modules (*The Classification module renames *Color by* to *Classify by* to form classification groups). For modules that come with point-based visualizations (e.g., scatter plot), hover-based interaction that show cell identifiers (red) and visual channel widgets S*hape by* and *Opacity by* derived from categorical features (blue) are implemented. Each module also comes with built-in modeling methods, supported by custom interactive widgets.

2.5.1 The Shared Interface

The shared interface supports common analysis tasks and is composed of interactive widgets built to operate the three feature classes—identifiers, numerical features, and categorical features (Supp Fig 6). The user has the option to map features in external datasets, i.e., datasets not output by Data Extraction, to these classes in the analysis configuration (Supp Fig 7). This option allows users to perform data extraction outside of FLIM Playground but still use the analysis methods provided by FLIM Playground.

Numerical features are grouped either by channel–extractor pairs or by custom groupings specified in the analysis configuration, making it faster to locate and select features. Univariate methods present one single-select widget per group; choosing a feature in any group clears the selections in the others. Bivariate methods present two such panels, with

the first chosen feature excluded from the second panel. Multivariate methods present one multi-select widget per group, with an *All* option to quickly include every feature in that group.

Filter widgets, each linked to a categorical feature, allow users to select one or more labels to find data of interest, with an *All* option available to include every label in that feature.

Human vision is optimized for rapid, parallel detection of patterns and trends; effective visual encodings—mappings of data to visual channels—leverage this capacity to reveal insights that raw numbers or text may obscure[27]. FLIM Playground provides visual channel widgets to map categorical features to colors, shapes, and opacities in the visualizations. Users can select multiple categorical features in the *Color by* widget; each unique label combination partitions the filtered dataset, with a distinct color assigned to each partition. For example, if the filtered dataset contains three treatments and two cell lines, and both features are selected in the *Color by* widget, six non-overlapping treatment–cell line partitions will be created. If no feature is selected, all data are grouped into a single partition and displayed in the same color. In point-based visualizations, the *Opacity by* and *Shape by* widgets allow a single categorical feature to be mapped so that each label is represented with a distinct opacity or shape. Interactive visualizations are implemented in Plotly[28], enabling users to hover over individual points (cells) to view their cell and field-of-view identifiers.

Widgets are also made to allow interactive adjustment of key visualization parameters, including point size in point-based visualizations, axis label size, legend size, and the colormap applied to distinguish groups.

2.5.2 Module-Specific Statistical Methods

Custom widgets are implemented to support module-specific modeling, and adjustments to them trigger real-time reanalysis. All models are applied separately to each partition created by the combinations of categorical features selected in *Color by*. These partitions will hereafter be referred to as color groups.

In *Feature Comparison* (Fig 4A-F), effect-size metrics—complementary to parametric tests—are calculated between comparison pairs, addressing the question of how large the difference is between two groups. FLIM Playground offers two effect size calculation methods: Glass' Delta[29] and Cohen's d[30].

Available comparison pairs include all pairwise combinations of color groups. Because the number of pairs can grow combinatorially and many may be uninformative, users can restrict the effect size annotations displayed in the visualization either by deselecting them in a selection widget or by setting a numerical threshold to show only annotations above a chosen value. The accompanied visualization comprises Sina[31] plots for each color group, revealing single-cell heterogeneity and group-level distributions of the selected numerical feature, with filtered effect size annotations overlaid. Users can designate a categorical feature in the *Separate by* widget to generate sub-panels corresponding to the labels of the selected feature. The *Color by* widget then creates color groups and comparisons are performed between pairs in each panel.

In *Feature Histogram* (Supp Fig 8), the selected numerical feature is visualized on the x-axis, and the y-axis represents either the counts or densities estimated by the Gaussian Mixture Model[32] (GMM) of each color group. The counts are determined by the user-adjustable bin width of the histograms. The default bin width is determined automatically by NumPy[33] as the minimum of the Sturges'[34] and Freedman–Diaconis[35] estimators.

As an alternative to histograms, Gaussian Mixture Models (GMMs) can capture multimodal distributions by representing a dataset as a weighted sum of Gaussian components, each with its component weights, means and variances. FLIM Playground uses scikit-learn[36] to fit all GMMs up to the user-controlled maximum number of components, retain only those in which all components exceed the user-specified minimum weight, and select the model with the lowest Bayesian Information Criterion[37] to prevent overfitting. To quantify the subpopulation structure of each color group from the GMM fit, a weighted entropy-based heterogeneity index is computed and displayed[38,39].

Each point can be assigned to a GMM component, which can then be exported and used as a categorical feature. In *Hard Assignment*, the posterior probability (responsibility) of each component is computed, and the data points are assigned to the component with the highest responsibility. Alternatively, *Intersection Thresholding* finds intersection points of adjacent component distributions using Brent's method[40] and assigns each point to the component whose region (bounded by these intersections) it falls into.

The bivariate *Feature Distribution* module (Supp Fig 9) visualizes the first selected feature on the x-axis and the second on the y-axis as a scatter plot. A two-dimensional GMM can be fit to each color group, following the same procedure as in *Feature Histogram*. The correlation coefficient and its p-value are calculated using SciPy[41], and a linear regression line can be fit using scikit-learn and displayed.

The *Phasor Analysis* module (Supp Fig 10) plots the phasor coordinates into the universal semicircle[25]. K-means clustering[42] is performed with a user-specified number of clusters, after standardizing the phasor coordinates since g (real part) and s (imaginary) may differ in scale. Then scikit-learn is used to find the optimized clustering. A convex hull, a polygon that contains all the points of each cluster, is drawn for each cluster, and the cluster centroid is marked with its coordinates displayed.

In *Dimension Reduction* (Fig 4G-I), selected numerical features are standardized and projected into two dimensions for visualization. Three popular algorithms are provided: Principal Component Analysis (PCA[43]), t-distributed Stochastic Neighbor Embedding (t-SNE[44]), and Uniform Manifold Approximation and Projection (UMAP[45]). Since hyperparameters affect projection results, users can adjust perplexity and early exaggeration when using t-SNE, while other parameters follow scikit-learn defaults. For UMAP, number of neighbors and minimum distance are adjustable, with remaining parameters set to default values of umap-learn.

In the *Classification* module (Fig 4H-M, Supp Fig 11), machine learning classifiers including Random Forest, Support Vector Machine, Logistic Regression, and Gradient Boosting are provided. Their performance is evaluated using accuracy, precision, recall, F1 score, and visual diagnostics including confusion matrices and Receiver Operating Characteristic (ROC) curves with corresponding Area Under the Curve (AUC). All

classifiers and performance metrics are implemented with scikit-learn, using a fixed random seed to ensure reproducibility. For decision tree–based models (Random Forest and Gradient Boosting), feature importance plots are also generated to rank the contribution of each feature.

A set of model-specific widgets are implemented to facilitate the classification process. Users can select the classifier and specify the proportion of data allocated to training and testing. To address class imbalance, three sampling strategies are provided. By default, stratified sampling that preserves the original class distribution is applied. Alternatively, undersampling[46] reduces all classes to the size of the smallest class, and oversampling increases all classes to the size of the largest class by randomly duplicating samples from minority classes. The visual channel widget *Color by* is renamed to *Classify by* and is used to form classes based on selected categorical features. Both binary and multi-class classification, augmented by one class versus the rest, options are available for user to select.

2.6 Two-photon fluorescence lifetime imaging microscopy of cells

PANC-1 human pancreatic (ATCC) and MCF7 human mammary gland (ATCC) adenocarcinoma cells were cultured in high-glucose Dulbecco's modified Eagle's medium supplemented with 10% fetal bovine serum (Gibco) and 1% penicillin/streptomycin (Gibco). 2×105 cells were plated on 35mm glass-bottom dishes (MatTek) 48 hours prior to imaging and maintained at 37°C and 5% CO2. The following metabolic inhibitors were added to the dish as indicated prior to imaging - sodium cyanide (NaCN; inhibits oxidative phosphorylation) (Fisher Scientific): 4mM for 15 minutes; 2-Deoxy-d-glucose (2-DG; inhibits glycolysis) (Sigma Aldrich): 10mM for 2 hours; iodoacetic acid (IAA; inhibits glycolysis) (Sigma Aldrich): 1.5mM for 30 mins. FLIM was performed on a custom-made Ultima Multiphoton Imaging System (Bruker) that consists of an inverted microscope (TI-E, Nikon). The system is coupled to an ultrafast tunable laser source (Insight DS+, Spectra Physics Inc). The fluorescence lifetime images were acquired using time-correlated single-photon counting (TSCPC) electronics (SPC -150, Becker & Hickl GmbH) and imaging was performed using Prairie View Software (Bruker). NAD(P)H and FAD were sequentially excited using excitation wavelengths of 750 nm and 890 nm respectively with the laser power at the sample <10 mW. The samples were illuminated using a 40x objective lens (W.I./1.15 NA /Nikon PlanApo) with a pixel dwell time of 4.8 μs and frame integration of 60s at 256×256 pixels. The photon count rates were maintained at 1–5 × 10^5 photons/second to ensure adequate photon detection for accurate lifetime decay fits. A 720nm dichroic mirror and bandpass filters separated fluorescence signals from the excitation laser. Emission filters used were bandpass 460/80 nm for NAD(P)H and 500/100 nm for FAD. Fluorescence signals were collected on GaAsP photomultiplier tubes (H7422P-40, Hamamatsu, Japan). The instrument response function (IRF) was collected each day by recording the second harmonic generation signal of urea crystals (Sigma-Aldrich) excited at 890nm. For each condition, data were acquired from 2 replicates and 3-4 images were acquired per replicate per condition.

2.7 SPCImage Analysis

To enable fitting using the SPCImage software (Becker & Hickl GmbH) at the cell-level, preprocessing was required: for each cell, pixel-level decays were aggregated by summing the decays of all pixels within the cell mask. The resulting cell-level decay was then propagated back to replace the decays of all pixels belonging to that cell, so that the pixel-fitting routine by SPCImage could be applied to fit cell-level decays[11,19]. Because all pixels of a given cell now shared the same decay curve, the binning factor was set to zero. The IRF time-shift was selected manually from one FOV by clicking on representative pixels containing cell decays; upon selection, SPCImage fits the decay and outputs the corresponding shift value. The median shift across those pixels was chosen and applied to all FOVs. The number of components was set to 2 to ensure bi-exponential fit.

3 RESULTS

3.1 Fit Validation

To validate the *Lifetime Fit* extractor in FLIM Playground, 25 NAD(P)H FLIM images of PANC-1 cancer cells were fitted using SPCImage and FLIM Playground. SPCImage yielded 6.40 as the IRF shift, while FLIM Playground produced 6.53. The cost metric (MLE), number of exponential components (2), and time gates were matched between the two methods to ensure a fair comparison. The average photon count across all cells was 4626. Lifetimes and fractions of each component were obtained from fits performed with these two methods.

The comparison of two representative lifetime features (τ_m, α_1) are shown in Fig 3. Although the mean of τ_m and α_1 calculated by FLIM Playground deviated from the SPCImage fit (Fig 3B, E) on average by -15.4 picoseconds (ps) [95% CI: -16.7 ps, -14.2 ps] and 0.18 % [95% CI: 0.1%, 0.2%] respectively, the effect sizes were small (Fig 3A, D), and the FLIM Playground values correlated consistently with the SPCImage values (r = 0.99 and p < 0.001; Fig 3C, F).

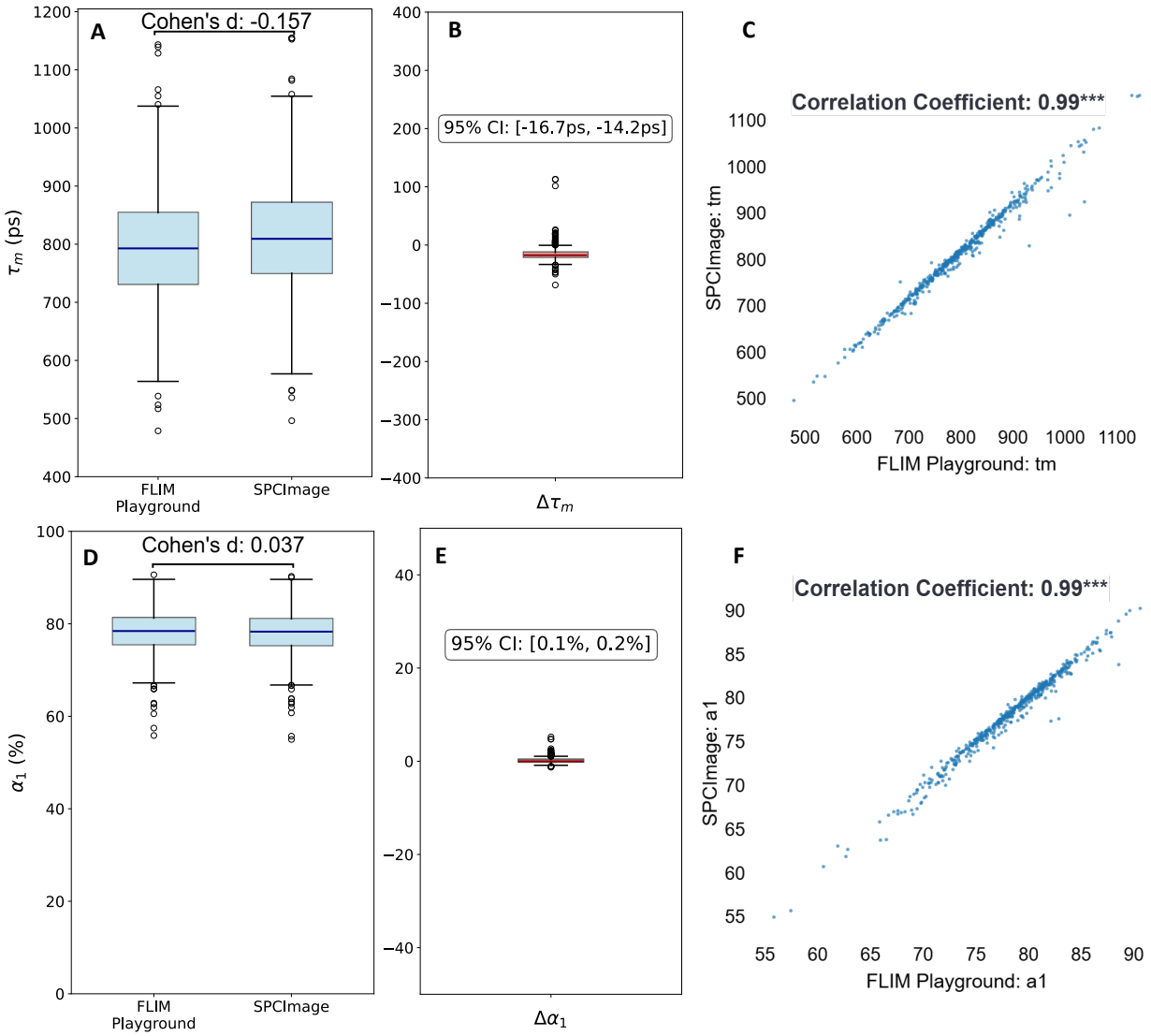

Fig 3: Fit Comparison between FLIM Playground and commercial software (SPCImage). The first row shows the comparison of the NAD(P)H mean lifetime (τ_m) calculated from FLIM Playground and SPCImage. **(A)** Distributions of τ_m are shown for both methods, with Cohen's d calculated between their means. **(B)** $\Delta\tau_m$ distribution is plotted on the same y-axis range as (A), with the 95% confidence interval annotated. **(C)** A scatter plot annotated with the correlation coefficient between the τ_m values obtained from the two fitting methods (p < 0.001), exported from the **Feature Distribution** module. **(D, E, F)** show the same visualizations and statistics for comparing the proportion of free NAD(P)H (α_1).

3.2 Discriminating Inhibitor treatments in PANC-1 and MCF7 Cells using FLIM Playground

Three inhibitors—cyanide, 2-DG, and IAA—were added to each of the two cell lines (PANC-1 and MCF7), alongside a control condition, and NAD(P)H and FAD channel FLIM images were acquired along with IRFs. Whole-cell masks were made using CellProfiler[13]

based on the NAD(P)H intensity images. Analysis was performed in FLIM Playground, including extracting single-cell features and visualizing and applying models to those features.

The settings were specified using the configuration interface (Supp Fig 1) and were kept consistent across subsequent stages. The FOV identification (Supp Fig 2) and IRF calibration (Supp Fig 3) stages were repeated for each day. Fitting was performed in Local mode, and a bi-exponential decay model was applied to both channels with the same cost metric (MLE) and the same time gates ([49, 240]). The fit lifetime features, phasor features, morphology and texture features were extracted for both channels (Supp Fig 4). All extracted features are shown in Supp Table 1.

The Data Extraction section was concluded by the categorical feature extraction stage, which merges the four single-cell numerical datasets and categories (cell line, treatment, and dish number) were interactively assigned to cells (Supp Fig 5).

The combined dataset was uploaded to the Data Analysis section, and a subset of methods was applied to gain biological insights (Fig 4). Through the shared interactive interface (Supp Fig 6), NAD(P)H τ_m was selected, categories of interest were selected (Fig 4A), and categorical features were assigned to visual channels (Fig 4B). Differences between groups were quantified using effect size calculations, where Glass' Delta was calculated for selected comparison pairs (Fig 4C), and an effect size threshold was chosen as 0.7 (Fig 4D). This produced a visualization showing the distributions of NAD(P)H τ_m and τ_ϕ for different inhibitors across two cell lines, with replicates encoded in different opacities (Fig 4E, F). Based on the large overlaps, technical replicate variability was relatively small. As expected, IAA increased the NAD(P)H τ_m and τ_ϕ compared to the control, whereas inhibition of oxidative phosphorylation by cyanide decreased NAD(P)H τ_m and τ_ϕ for both cell lines (Fig. 4E,F).

The UMAP dimensionality reduction functionality (Fig. 4G, H) was demonstrated on PANC-1 cells, using all features (Supp Table 1) except single-cell intensity sum to ensure independence from acquisition-related intensity differences. Separation between treatments was clear: the two glycolysis inhibitors (2DG, IAA) clustered on one side while the oxidative phosphorylation inhibitor (cyanide) localized to the opposite region of the embedding space, and the control lay in the middle (Fig. 4H). PCA (Fig 4I) was demonstrated on MCF7 cells using the same features as UMAP. The separation between control and 2DG was less distinct, consistent with the small effect size observed between these groups for both τ_m and τ_ϕ.

Whereas dimension reduction methods provided a qualitative view of treatment separation, classification was used to quantify the discriminative power of the features. To discriminate between the cyanide, the oxidative phosphorylation inhibitor, and the glycolysis inhibitors (2DG and IAA), "cyanide versus the rest" was selected in the classification options list (Fig 4J). The train-test split was 70% and 30% (Fig 4K). Since the classes (cyanide vs 2DG + IAA) were not substantially imbalanced, the default stratified sampling was used. All classifiers achieved an AUC of 1.0 on the ROC curves (Fig 4L), with the near diagonal confusion matrix (of the Random Forest classifier, Fig 4M), indicating strong discriminative ability between cyanide vs. the glycolysis inhibitors (2DG and IAA).

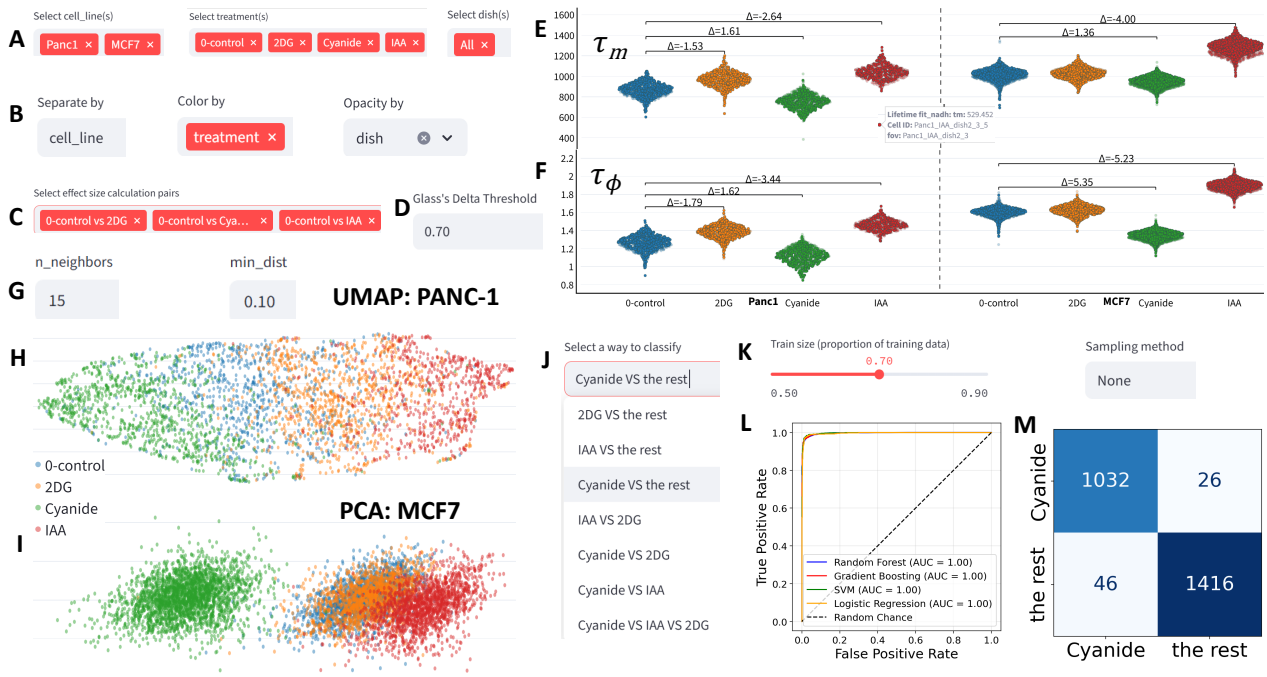

Fig 4: Selected interactive widgets and analysis results. (A) Filter widgets are rendered for each categorical feature. Two cell lines, three inhibitors plus the control, and both dishes were selected. **(B)** Data separated by cell line, each generating one sub-panel; within each panel, treatments are color-coded. Different opacities indicate variability between dishes. **(C)** Three comparison pairs, control versus each inhibitor, were selected, and **(D)** only effect sizes exceeding the user-specified threshold (0.7) were shown. **(E)** NAD(P)H τ_m was compared between the inhibitors and the control, separated by cell line (left: PANC-1, right: MCF7). Users can hover over each point to see its identifiers. **(F)** Same comparisons plotted for τ_ϕ, a phasor derived lifetime feature. **(G)** Interactive widgets for specifying UMAP hyperparameters (*n_neighbors* = 15, *min_dist* = 0.1); **(H)** UMAP was performed on PANC-1 cells; **(I)** PCA of MCF7 cells; **(J)** List of all classification options; cyanide versus the other two glycolysis inhibitors (2DG, IAA) was selected. **(K)** Train–test split: 70% (5,877 cells) training and 30% (2,520 cells) testing and the sampling method used (None: default stratified sampling). **(L)** ROC curves generated from FLIM Playground classifiers. All classifiers were trained on the same numerical feature set used for UMAP. **(M)** Confusion matrix of the Random Forest classifier evaluated on unseen data, showing True Positives (upper left), False Negatives (upper right), False Positives (lower left), and True Negatives (lower right).

4. DISCUSSION

FLIM Playground is an open-source, interactive platform that unifies metadata organization, calibration, single-cell feature extraction, visualization, and statistical modeling through an intuitive and code-free graphical interface. A diverse set of tools offering alternative methods exists, but switching between them and processing their outputs to obtain biologically interpretable features and insights still requires custom code and expertise. FLIM Playground overcomes this limitation by focusing on three semantically meaningful feature classes—identifiers, numerical features, and categorical features—which align closely with the hierarchical levels of FLIM data (Table 1).

The Data Extraction section adopts an acquisition channel–centric framework, in which each channel can be assigned its own extraction method to allow flexibility for various input types. Notably, extraction is not limited to lifetime features: morphological and texture descriptors are also supported, enabling richer characterization of cellular

phenotypes. In addition to numerical features, identifiers and categorical features are also incorporated into a three-stage seamless extraction interface. The Data Analysis section transforms each of the three extracted feature classes into a unified interface, shared across visualization and analysis methods built on the Python ecosystem. This integration allows users to move beyond extraction toward analyses that capture biologically meaningful variation across cells in a single platform. To demonstrate this integration and its capabilities, FLIM Playground was used to run the complete pipeline—from feature extraction through analysis—on representative two-photon NAD(P)H and FAD FLIM images of cancer cell lines (Panc1 and MCF7) treated with metabolic inhibitors, along with their corresponding single-cell masks generated in CellPose. To ensure reliability, lifetime fits in FLIM Playground were benchmarked against the widely used commercial package SPCImage.

While FLIM Playground provides an integrated framework for extraction and analysis, several limitations remain. At present, the software is designed for time-correlated single-photon counting (TCSPC)-acquired data; however, support for other time-domain and frequency-domain acquisition modalities could be implemented in future versions. Similarly, additional fit-free methods, such as Laguerre deconvolution[26], may be incorporated to broaden the feature extraction repertoire. Fitting is currently limited to the cell level due to computational speed constraints, making pixel-level fitting impractical, though FLIM Playground can accept pixel-level prefit lifetime features or allow users to perform cell-level fitting. Although pixel-level phasor calculation, visualization, and texture descriptors of phasor feature maps are technically feasible, they have not yet been implemented. The extraction module can also be expanded to include a wider range of morphological and texture feature descriptors. Finally, the current version does not support time-lapse FLIM imaging or object tracking. Numerical features that capture relationships across time, such as velocity, and their correlation with lifetime features would provide valuable insights and will be considered in future developments.

FLIM Playground makes advanced FLIM analysis accessible to non-computational practitioners through its intuitive interface, while its modular design facilitates improvements by specialists, enabling the framework to be readily adapted for non-FLIM acquisitions as well. The platform can be expanded to incorporate additional acquisition modalities, feature descriptors, and fit-free methods, supporting a range of applications in quantitative imaging. By unifying extraction, visualization, and analysis in a single platform, FLIM Playground supports biologically meaningful discoveries at the single-cell level and at scale.

Acknowledgements

The authors would like to thank all members of the Skala Lab for their suggestions, as well as those who kindly proofread the online manual. We are also grateful to Matthew Stefely for designing and creating the FLIM Playground logo.

Funding

This work was funded by
NIH R01 CA278051,
R01 CA272855,

R01 HL165726,
R37 CA226526,
NSF 2426316

Author Contributions

Conceptualization: WZ, KS, MCS, RD*,
Methodology: WZ, KS
Investigation: WZ, RD*
Formal analysis: WZ, KS
Funding acquisition: MCS
Writing – Original Draft Preparation: WZ, RD*
Writing – Review & Editing: WZ, KS, MCS, RD*
*corresponding author

Competing Interests

MCS is an adviser to Elephas Biosciences. All other authors declare they have no competing interests.

Code and Data Availability

The source code is available upon request and will be released as open source upon publication. The dataset used and the software executables are available at https://github.com/skalalab/flim_playground_bioarchive.

Reference:

1. Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. Cell Detection with Star-Convex Polygons. in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018* 265–273 (Springer, Cham, 2018). doi:10.1007/978-3-030-00934-2_30.

2. Stringer, C. & Pachitariu, M. Cellpose3: one-click image restoration for improved cellular segmentation. *Nat Methods* **22**, 592–599 (2025).

3. Sofroniew, N. *et al.* napari: a multi-dimensional image viewer for Python. Zenodo https://doi.org/10.5281/zenodo.16883660 (2025).

4. Riendeau, J. M. *et al.* Cellpose as a reliable method for single-cell segmentation of autofluorescence microscopy images. *Sci Rep* **15**, 5548 (2025).

5. Gao, D. *et al.* FLIMJ: An open-source ImageJ toolkit for fluorescence lifetime image data analysis. *PLoS One* **15**, e0238327 (2020).

6. Becker, D. W. *The Bh TCSPC Handbook: Ninth Edition*. (Becker & Hickl GmbH, 2021).

7. Ranjit, S., Malacrida, L., Jameson, D. M. & Gratton, E. Fit-free analysis of fluorescence lifetime imaging data using the phasor approach. *Nat Protoc* **13**, 1979–2004 (2018).

8. Gottlieb, D., Asadipour, B., Kostina, P., Ung, T. P. L. & Stringari, C. FLUTE: A Python GUI for interactive phasor analysis of FLIM data. *Biological Imaging* **3**, e21 (2023).

9. Gohlke, C., Pannunzio, B., Schüty, B. & Blanco, R. phasorpy/phasorpy: v0.7. Zenodo https://doi.org/10.5281/zenodo.16923774 (2025).

10. Samimi, K. *et al.* Autofluorescence lifetime flow cytometry with time-correlated single photon counting. *Cytometry Part A* **105**, 607–620 (2024).

11. Samimi, K., Desa, D. E., Pham, D. L. & Skala, M. C. ROI-summed analysis for improved fluorescence lifetime fitting accuracy and precision. in *Multiphoton Microscopy in the Biomedical Sciences XXV* vol. PC13324 PC1332412 (SPIE, 2025).

12. Kapsiani, S. *et al.* FLIMPA: A Versatile Software for Fluorescence Lifetime Imaging Microscopy Phasor Analysis. *Anal. Chem.* **97**, 11382–11387 (2025).

13. Stirling, D. R. *et al.* CellProfiler 4: improvements in speed, utility and usability. *BMC Bioinformatics* **22**, 433 (2021).

14. Streamlit • A faster way to build and share data apps. https://streamlit.io/ (2021).

15. pyinstaller/pyinstaller: Freeze (package) Python programs into stand-alone executables. https://github.com/pyinstaller/pyinstaller.

16. skalalab/flim_playground_bioarchive: Data and executables for FLIM Playground. *GitHub* https://github.com/skalalab/flim_playground_bioarchive.

17. Zhao, W. FLIM Playground. https://allanware.quarto.pub/flim-playground-doc/ (2025).

18. Ranjit, S., Malacrida, L., Jameson, D. M. & Gratton, E. Fit-free analysis of fluorescence lifetime imaging data using the phasor approach. *Nat Protoc* **13**, 1979–2004 (2018).

19. Gohlke, C. cgohlke/sdtfile: v2025.3.25. Zenodo https://doi.org/10.5281/zenodo.15086209 (2025).

20. Gohlke, C. cgohlke/ptufile: v2025.5.10. Zenodo https://doi.org/10.5281/zenodo.15380998 (2025).

21. Newville, M. *et al.* LMFIT: Non-Linear Least-Squares Minimization and Curve-Fitting for Python. Zenodo https://doi.org/10.5281/zenodo.16175987 (2025).

22. Storn, R. & Price, K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization* **11**, 341–359 (1997).

23. Marquardt, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics* **11**, 431–441 (1963).

24. Nelder, J. A. & Mead, R. A Simplex Method for Function Minimization. *Comput J* **7**, 308–313 (1965).

25. Digman, M. A., Caiolfa, V. R., Zamai, M. & Gratton, E. The Phasor Approach to Fluorescence Lifetime Imaging Analysis. *Biophysical Journal* **94**, L14–L16 (2008).

26. Datta, R., Heaster, T. M., Sharick, J. T., Gillette, A. A. & Skala, M. C. Fluorescence lifetime imaging microscopy: fundamentals and advances in instrumentation, analysis, and applications. *JBO* **25**, 071203 (2020).

27. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods on JSTOR. https://www-jstor-org.ezproxy.library.wisc.edu/stable/2288400.

28. plotly/plotly.py: The interactive graphing library for Python :sparkles: https://github.com/plotly/plotly.py.

29. Glass, G. V. Primary, Secondary, and Meta-Analysis of Research. *Educational Researcher* **5**, 3–8 (1976).

30. Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*. (Routledge, New York, 2013). doi:10.4324/9780203771587.

31. Sidiropoulos, N. *et al.* SinaPlot: An Enhanced Chart for Simple and Truthful Representation of Single Observations Over Multiple Classes. *Journal of Computational and Graphical Statistics* **27**, 673–676 (2018).

32. Dempster, A. P., Laird, N. M. & Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39**, 1–38 (1977).

33. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585**, 357–362 (2020).

34. Sturges, H. A. The Choice of a Class Interval. *Journal of the American Statistical Association* **21**, 65–66 (1926).

35. Freedman, D. & Diaconis, P. On the histogram as a density estimator:L 2 theory. *Z. Wahrscheinlichkeitstheorie verw Gebiete* **57**, 453–476 (1981).

36. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).

37. Schwarz, G. Estimating the Dimension of a Model. *The Annals of Statistics* **6**, 461–464 (1978).

38. Sharick, J. T. *et al.* Metabolic Heterogeneity in Patient Tumor-Derived Organoids by Primary Site and Drug Treatment. *Front. Oncol.* **10**, (2020).

39. Datta, R. *et al.* Single cell autofluorescence imaging reveals immediate metabolic shifts of neutrophils with activation across biological systems. *Front. Immunol.* **16**, (2025).

40. Brent, R. P. & Brent, R. P. ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES.

41. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods* **17**, 261–272 (2020).

42. Lloyd, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory* **28**, 129–137 (1982).

43. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**, 559–572 (1901).

44. Maaten, L. van der & Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research* **9**, 2579–2605 (2008).

45. McInnes, L., Healy, J., Saul, N. & Großberger, L. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software* **3**, 861 (2018).

46. Lemaître, G., Nogueira, F. & Aridas, C. K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* **18**, 1–5 (2017).