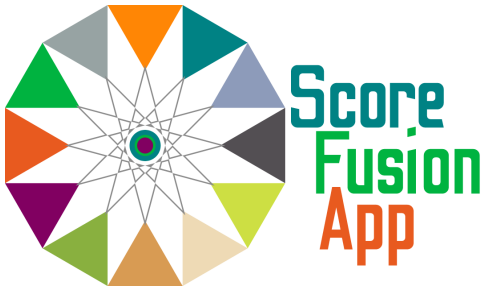


Score Fusion App

Guide to Fusing Multibiometric Scores.

User Manual

Version: 1.0

Author
Melissa R Dale

Supervisor
Arun Ross

August 25, 2021

Contents

Chapter 1

Introduction

The Score Fusion App was designed to facilitate experiments on multibiometric score data. This manual is intended to provide technical details on how to use and adapt this software application. To learn more about the science of multibiometrics, please consult [1].

The application loads biometric score data in a variety of file formats, provides data grooming, visualizations, and metric descriptors of the loaded scores, in addition to providing a variety of fusion rules to analyze. The following chapters of this manual describe how to acquire the app, what score files should look like, and how to customize fusion experiments. In the Advanced chapter, the software itself is described in detail to facilitate the addition of new fusion rules and customization options.

1.1 Notes on Terminology

Biometric Match Scores A typical biometric system compares the *probe* data with the identities stored in a *gallery* database of user templates [1]. To accomplish this, the feature set extracted from the probe is compared against each gallery template resulting in a match **score**. These match scores are then sorted to determine those gallery identities with the highest degree of similarity to the probe.

The Genuine and Imposter Class. The application uses the terms Genuine and imposter throughout the app and the user manual. A genuine score indicates a score that

Verification vs Identification. *Verification* tasks aim to verify a claimed idea. The output of the system is binary: Genuine or Imposter. *Identification* tasks attempt to identity a user from the system's enrolled users. The output of an identification task is the identity of the probe.

Chapter 2

Using the Score Fusion App

2.1 Quick Start

2.1.1 Executable Download

Executable versions of the Score Fusion App can be downloaded from the releases directory (https://github.com/melissadale/ScoreFusionApp/tree/master/Releases). Work is ongoing to increase stability of the portable executable. If it does not currently work on the user's machine, please consider compiling from source.

2.1.2 Download from Source (GitHub)

If you would like to adapt the application, or if the packaged executable does not work for you, you can set up the application by following this section.

Prerequisites

- **Conda**: Conda creates a virtual environment which facilitates acquiring the correct python packages and the correct versions of those packages.
- **git:** Git allows you to collect the code from GitHub.

Once the prerequisites are installed, you can run the .sh script provided in the Score Fusion App's GitHub page to check out the code as well as install necessary packages to a Conda environment.

If you are unable to run the .sh script, you can manually run each of the following commands from your terminal or powershell, and navigate to the directory you'd like the score fusion app code to reside).

```
conda create --name scorefusion_environment python=3.8.5
conda activate scorefusion_environment

git clone https://github.com/melissadale/ScoreFusionApp.git

pip install  -r $PWD/ScoreFusionApp/Utilities/requirements.txt
sudo apt-get install libmtdev1
```

You can now edit the score fusion app code or run the application through your favorite editor, like PyCharm after configuring the python interpreter to be the scorefusion_environment conda environment.

If assistance is needed, please consult the Feature Requests and Support chapter.

2.2 Input Files

This application accepts score data in 2 formats, **matrix** format or **column** formats. A matrix format assumes that each modality's scores are saved in a separate file (either csv or txt file), with the genuine scores along the diagonal. A column format contains all the score data in a singular file. The details and differences of these formats are explored in the following subsections, and summarized in Figure 2.1.

The user does not need to specify which format the data is in, the application will automatically detect this information.

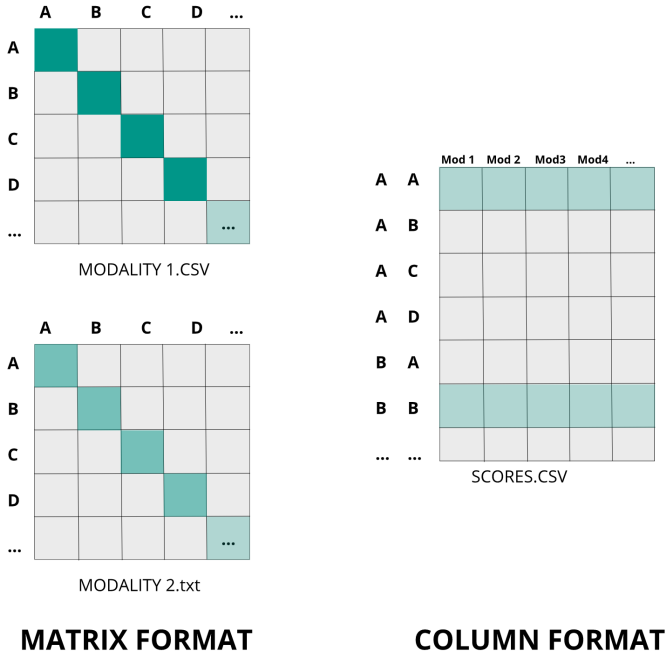


Figure 2.1: Expected input file format: Matrix Form (Left) and Column Form (Right)

Things that are true, regardless of input file format:

- **Subject IDs**: The application does its best to identify what might be a subject id. IF IDs are strictly numeric, that makes detection more challenging. If Subject IDs are included as column or row headers, ensure they are not strictly numeric.
- **Test / Train detection**: The application checks filenames for "test" and "train" to determine if the data is already split into training and testing samples. Please be sure to have train and test anywhere in the filename where appropriate.

2.2.1 Matrix Form

This score format contains subjects along the rows and columns such that genuine scores are along the diagonal, and the impostor scores are off diagonal.

Good things to know:

- labels are not necessary, scores along the diagonal are *genuine*, off diagonal scores are *impostor*
- Each modality should be its own file. That file name will be what the modality is referred to as in the application.

A pictorial example of this format for the NIST Biometric Score Dataset (Face x Finger) at NIST BSSR1, with subject IDs as row and column headers.

	A	B	C	D	E	F	G
1		0010B559B76132149D909F438AD0CA8E	00D5B283	00DCA46A	00FF2F1FE	019395975	02D539514
2	0010B559B76132149D909F438AD0CA8E	73.59216	65.33185	66.44193	65.66969	64.88933	66.14874
3	00D5B283FBC05619C124CCE48F35F181	73.85016	77.67635	65.84585	65.88991	66.77268	70.96465
4	00DCA46A1BF566A2F23C0525B434DE76	65.70415	65.25148	79.65856	66.44344	66.71967	64.91592
5	00FF2F1FBAE0F6124E4EF77C7B519784	65.41616	65.78752	66.07175	79.47175	63.79398	65.24973
6	01939597540A36FE952EF0CEFC74ADCE	65.89835	71.91535	63.93061	62.82392	65.93723	65.52065
7	02D5395147275449C5D903C137E88320	65.64023	70.40938	65.77727	65.03819	70.18008	77.31303
8	030131B827536FED9F91818123D6EFE0	66.66251	66.61274	71.58596	66.41062	66.46071	70.60197
9	03866493F8412FCC73F584891B3FA0EB	64.86715	65.70439	66.16643	66.188	66.21091	64.13432
10	0442FDBDE8BFF8E47C385640FC7D0CC1	66.17557	71.17485	64.49448	63.94925	66.56596	65.78635
11	045189AB35312C12DD16DF1033253A41	65.19192	71.66972	65.3483	65.37913	65.79822	64.50978
12	05C96477B6CC75E183B0D9AFD5925265	65.39006	65.57166	66.03555	72.49865	65.94054	65.81306
13	06CF27CDCC4E54F2E6F97A87C4D1C280	66.10124	66.99481	66.11432	65.62312	70.93041	65.16029
14	072CF49EFD92B5A2F8B2718CE8DFAFD5	66.17219	72.93562	64.78555	65.06091	66.0259	64.36529

Figure 2.2: Score Matrix with Headers.

Assumptions:

- The order of subjects is consistent along the column and rows
- There are the same number of rows and columns
- IF there are row or column headers identifying the subject id, that id is not in the same format as the scores (i.e. subject ids should not be floats or pure integers)

2.2.2 Column Form

Column formatted data are assumed to be represented as a collection of columns, which contain ***at least***:

- Subject ID
- Gallery ID
- Score Column(s)

$N : M$ Score Data

If there are more than $1 : N$ scores for subject to gallery ids, please ensure that the score data is formatted such that the following column headers exist:

- probe_subject_id
- probe_file_id
- candidate_subject_id

- candidate_file_id
- genuine_flag

Important things to know:

This format is based along the concept that there are modalities along the columns, with the last column containing the label(0 impostor, 1 genuine) OR the first 2 columns must be subject ids - in which case, genuine impostor labels may be determined by checking if the first two column ids are equal (genuine) or not (imposter).

- Each column must have a header. This is how the application knows modality titles.
- The label for this last column should be titled label or class (capitalization does not matter)

2.3 Missing Scores

The Score Fusion App provides multiple options on how to deal with missing scores. Many multimodal biometric datasets are complete, meaning that each probe contains a score for each gallery identity. In this case, there is no need to address the issue of missing scores (e.g. ignore missing scores). However, in many real world use cases, there are many situations where a probe may not be able to return the scores for each identity in the gallery. This can lead to a sparse dataframe.

In these circumstances, it is necessary to either drop incomplete score vectors, or fill empty spaces with a proxy value. The following two subsections define the options for handling missing score values.

2.3.1 Ignore Missing Scores

The default option is to ignore missing scores. If this option is selected, incomplete score vectors are simple dropped. If there are a small number of missing scores, or no missing scores, use this option.

Select Ignore if:

- The score data is complete and does not contain any missing scores.
- The number of missing scores is small

2.3.2 Impute Missing Scores

If the data contains more than a few missing scores, you can choose to impute missing score values. The app supports a variety of imputation methods:

- **Mean and Median**: These two options are univariate and straightforward. A modality's missing scores are filled with that modality's mean or median score.
- **Multiple Imputation by Chained Equations (MICE)**: MICE approach to imputations uses a machine learning model to iteratively update missing values by initially filling missing values with the mean or median score value described above,

and then iteratively updating the placeholder values. The following approaches are available in the app:

- **Bayesian Ridge Regression (Linear)**
- **Decision Tree Regression (Non-Linear)**
- **K Nearest Neighbors Regression** Note, this option often returns good results, however can also be prohibitively time expensive to run.

2.4 Train and Test Splitting

2.5 Normalization

Normalization puts all the scores into a common domain. Consider a matcher's whose scores are between -1 and 1, while another matcher's scores are between 100 and 500. In order to fuse the scores from these two matchers, the scores must be normalized into a common range.

The following are the normalization techniques supported by the Score Fusion App.

2.5.1 MinMax

MinMax normalization scales score values between 0 and 1, and it is a common normalization approach. This is the Score Fusion App's default normalization. MinMax normalization is sensitive to outliers.

$$s' = \frac{s - M_{min}}{M_{max} - M_{min}} \tag{2.1}$$

2.5.2 Z-Score

Z-Score normalization is more robust to outliers by using scores' means (μ) and standard deviations (σ). Note that if the unnormalized data has a large standard deviation, the normalized score values will be near 0.

$$s' = \frac{s - M_{\mu}}{M_{\sigma}} \tag{2.2}$$

2.5.3 Decimal

Decimal scaling puts scores into the range of 0 to 1. For instance, consider scores in the range 100 to 9,000. These are scaled to the range 0.01 to 0.9 by dividing score s by 10^4. Equation 2.3 formally defines decimal scaling. In the above example, $j = 4$, as the largest score value is 9,000 (10^4).

$$s' = \frac{s}{10^j} \tag{2.3}$$

2.5.4 Median & MAD

$$s' = \frac{s - median(s)}{MAD(s)} \tag{2.4}$$

2.5.5 Double Sigmoid

$$s' = \begin{cases} 1/(1 + (exp(-2((s-t)/r_1))), & \text{if s} < \text{t} \\ 1/(1 + (exp(-2((s-t)/r_2))), & \text{if s} \geq \text{t} \end{cases} \tag{2.5}$$

2.5.6 Tanh Estimator

$$s' = \frac{s - median(s)}{MAD(s)} \tag{2.6}$$

2.5.7 None

You can choose to not normalize the scores to a common domain. This option is not recommended unless you are certain that all modalities already are in a consistent range.

$$s' = s \tag{2.7}$$

2.6 Editing Detected Modalities

Modalities are automatically detected by either the file names or column headers, and by default are set as similarity scores. Users can edit these by clicking on the pencil icon next to the "Detected Modalities" pane. This will bring up the "Edit Modalities" popup as seen in Figure 2.3.

This popup allows users to change the names of the modalities, select a subsection of available modalities for fusion, and specify if a modality's scores are dissimilarity scores. Note, the Update button must be clicked for these changes to take effect.

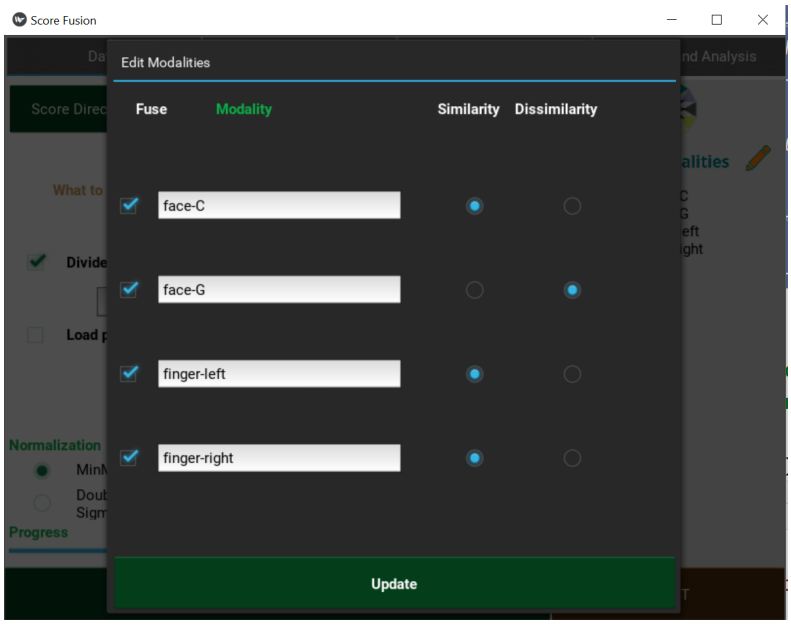


Figure 2.3: Edit detected modalities screen.

2.7 Density Estimates

The Density Estimates Tab features descriptions of the score data. This includes estimated Probability Density Functions (PDFs) and histograms of the genuine and imposter class scores for each of the detected modalities. These visualizations, as well as metrics such as the numbers of genuine and imposter users in the entire dataset, the training dataset, and the testing dataset for each modality.

Figure 2.4 provides a screenshot of this panel.

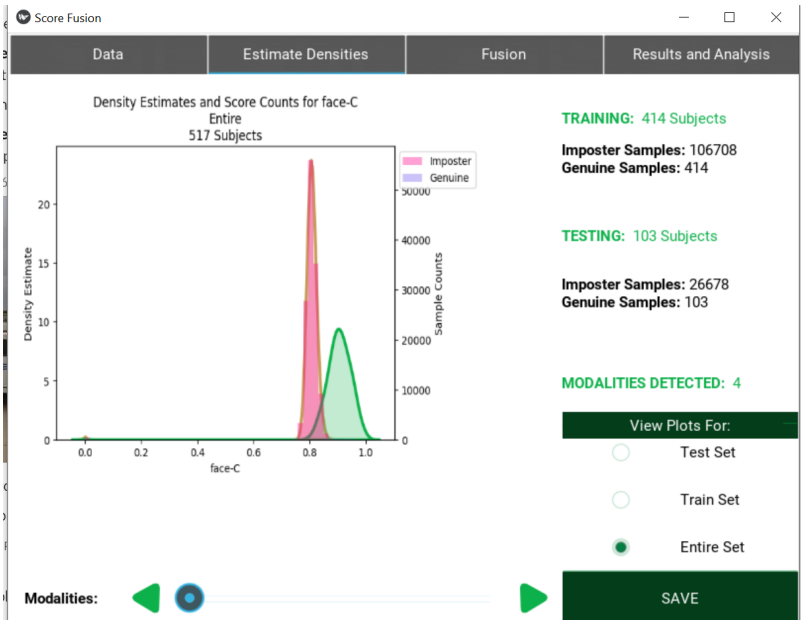


Figure 2.4: Estimate Densities tab, which features score distribution plots and test/train metrics for modalities.

2.8 Fusion

The Fusion Panel is where the user defines the experiments they wish to run by checking which fusion rules to apply, give the experiment names, and specify the type of task (verification and/or identification).

This panel is shown in Figure 2.5.

2.8.1 Sequential Fusion Rule

If the Sequential Fusion is selected, a popup will present the user with a screen to define the baseline modality, as well as an option to automatically detect or manually define score ranges that indicate contentious scores.

This popup is shown in Figure 2.6.

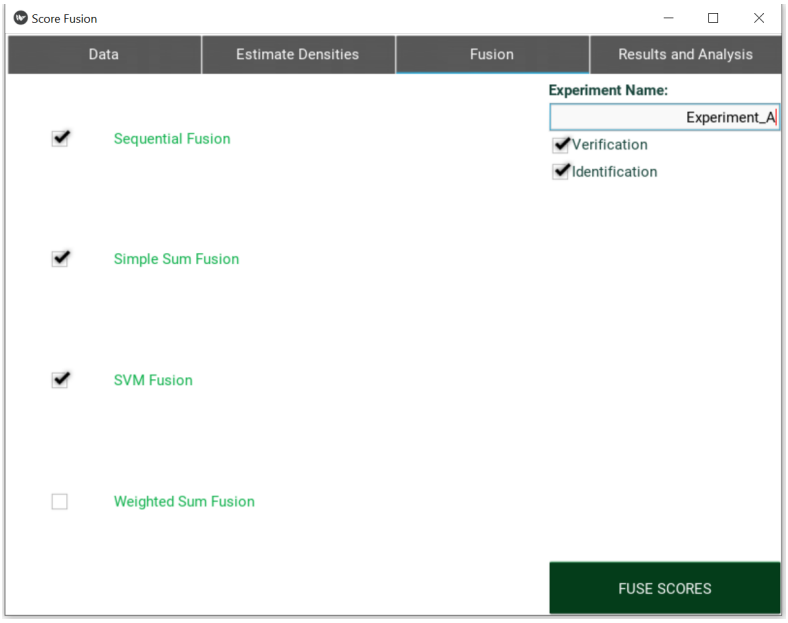


Figure 2.5: Screenshot of Fusion panel, where fusion rules and experiments are defined.

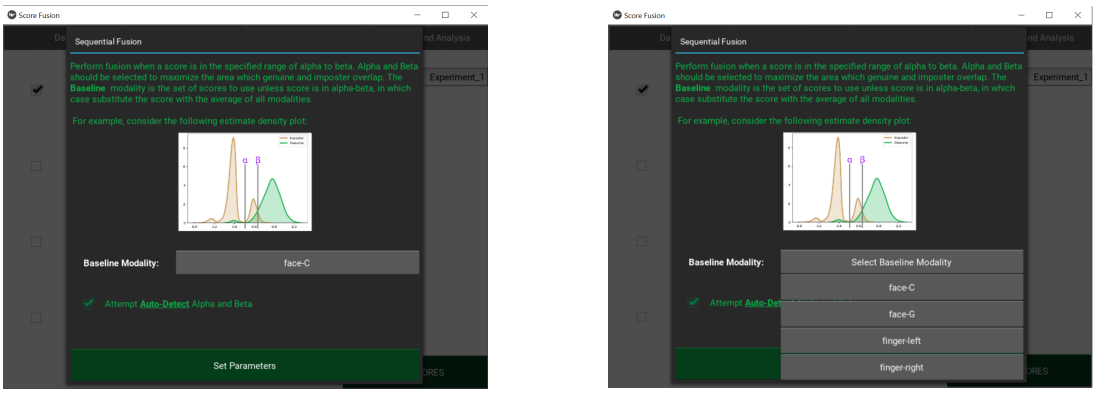


Figure 2.6: Sequential Fusion Popup

2.9 Experiments and Results

The Experiment and Results panel display the experiment's results, including plots and numeric metrics. On the bottom of the panel is a slider, that allows the user to traverse named experiments.

The verification task experiments show ROC plots, AUC accuracy, Equal Error Rate, and Estimated TMR values for a user defined Fixed FMR value. The ROC plots are click through. The first plot contains the ROC curves for all modalities and fusion rules, each fusion rule compared to the modalities, and only the modalities.

The Identification task experiments show CMC plots, Rank 1 and Rank 2 accuracies, as well as the option for a user to select Rank K, where K<20. The CMC plots are click through. The first plot contains the CMC curves for all modalities and fusion rules, the baseline modalities, and the fusion rules.

The identification and verification task panels are shown in Figure 2.7.

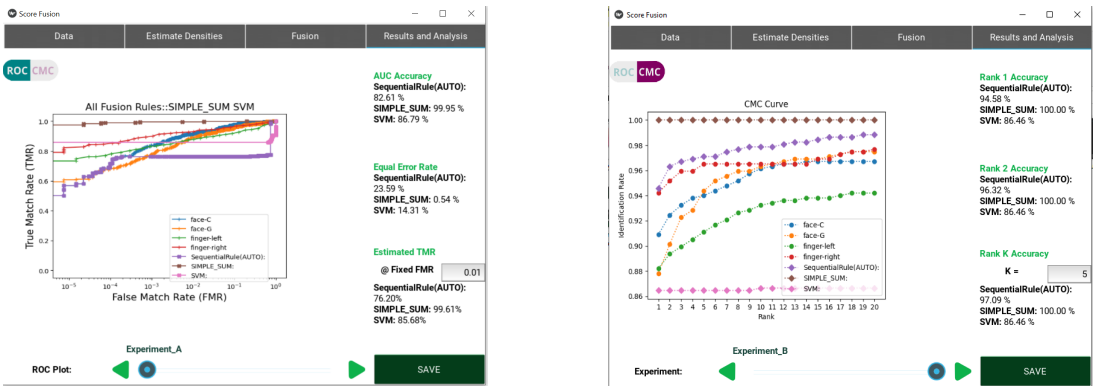


Figure 2.7: Identification Task Results Panel (Left) and Verification Task Results Panel (Right)

2.10 Saving Experiment Results to Report

The Score Fusion App can save the data displayed in the results panel to a PDF report. The following pages provide examples of the reports generated for verification and identification tasks.

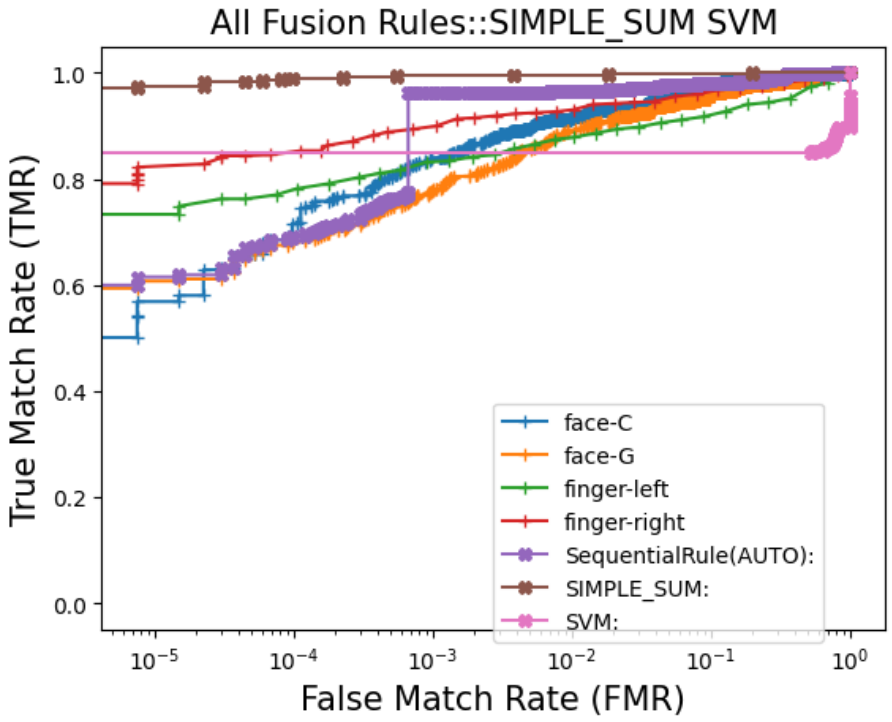
Summary of Score Fusion Experiments

Generated on: 2021-08-25

Experiment:

Fusion Rules Applied: SequentialRule(AUTO), SIMPLE_SUM, SVM

Modalities: face-C, face-G, finger-left, finger-right



Fusion Rule	AUC	EER	TMR @0.01FMR
face-C	0.98898	0.04449	0.91489
face-G	0.98288	0.05803	0.89168
finger-left	0.96265	0.08393	0.87427
finger-right	0.98233	0.04865	0.9265
SequentialRule(AUTO):	0.99266	0.02901	0.96518

Summary of Score Fusion Experiments

Generated on:	2021-08-25		
SIMPLE_SUM:	0.99957	0.00387	0.99613
SVM:	0.86136	0.15087	0.84913

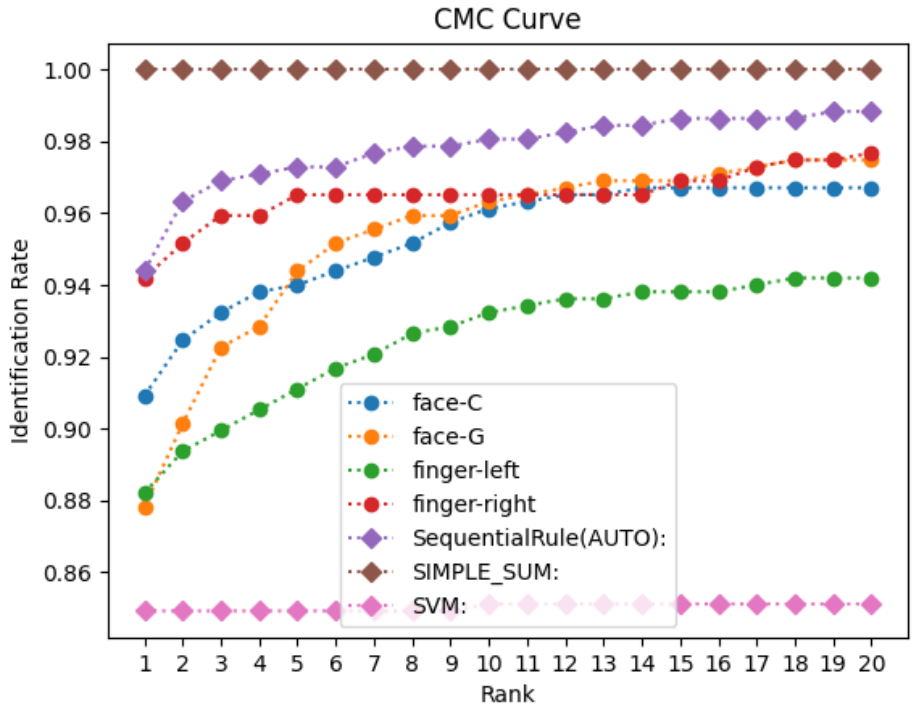
Summary of Score Fusion Experiments

Generated on: 2021-08-25

Experiment:

Fusion Rules Applied: SequentialRule(AUTO), SIMPLE_SUM, SVM

Modalities: face-C, face-G, finger-left, finger-right



Fusion Rule	Rank 1	Rank 2	Rank 5
face-C	0.90909	0.92456	0.94004
face-G	0.87814	0.90135	0.94391
finger-left	0.88201	0.89362	0.91103
finger-right	0.94197	0.95164	0.96518
SequentialRule(AUTO):	0.94391	0.96325	0.97292
SIMPLE_SUM:	1.0	1.0	1.0

Summary of Score Fusion Experiments

Generated on:	2021-08-25		
SVM:	0.84913	0.84913	0.84913

Chapter 3

Advanced

This chapter is for the advanced user of Score Fusion App, and is intended to help others adapt the code for their own purposes. This section focuses on the software design and naming conventions.

3.1 Internal Score Data Structure

No matter how the input files are structured, this app creates a consistent score matrix representation to be used throughout interactions with the application. This score matrix dataframe contains the following columns: Train_Test category, Probe_ID, Gallery_ID, Label, Original Scores (multiple columns), Normalized Scores (multiple columns), and Fused Scores (one or more multiple columns).

For Example, Figure 3.1 is a screenshot of the internal score matrix generated for the NIST BSSR1 multimodal dataset.

	Train_Test	Probe_ID	Gallery_ID	Label	face-C_ORIGINAL	face-G_ORIGINAL	finger-left_ORIGINAL	finger-right_ORIGINAL	face-C	face-G	finger-left	finger-right
0	TRAIN	0010B559B76132149D909F438AD0CA8E	0010B559B76132149D909F438AD0CA8E	1.00000	0.57561	73.59216	29.00000	84.00000	0.83546	0.65449	0.11789	0.32685
1	TEST	00D5B283FBC05619C124CCE48F35F181	00D5B283FBC05619C124CCE48F35F181	1.00000	0.78525	77.67635	6.00000	57.00000	0.94662	0.79700	0.02439	0.22179
2	TRAIN	00DCA46A1BF566A2F23C0525B434DE76	00DCA46A1BF566A2F23C0525B434DE76	1.00000	0.81421	79.65856	63.00000	81.00000	0.96197	0.86617	0.25610	0.31518
3	TRAIN	00FF2F1FBAE0F6124E4EF77C7B519784	00FF2F1FBAE0F6124E4EF77C7B519784	1.00000	0.82965	79.47175	73.00000	65.00000	0.97016	0.85965	0.29675	0.25292
4	TRAIN	01939597540A36FE952EF0CEFC74ADCE	01939597540A36FE952EF0CEFC74ADCE	1.00000	0.59057	65.93723	175.00000	158.00000	0.84339	0.38738	0.71138	0.61479
5	TRAIN	02D53951472754495C5D903C137E88320	02D53951472754495C5D903C137E88320	1.00000	0.67594	77.31303	10.00000	25.00000	0.88865	0.78432	0.04065	0.09728
6	TRAIN	030131B827536FED9F91818123D6EFE0	030131B827536FED9F91818123D6EFE0	1.00000	0.67123	75.71003	11.00000	17.00000	0.88616	0.72839	0.04472	0.06615
7	TRAIN	03866493F8412FCC73F584891B3FA0EB	03866493F8412FCC73F584891B3FA0EB	1.00000	0.77887	78.02557	38.00000	130.00000	0.94323	0.80919	0.15447	0.50584
8	TRAIN	0442FDBDE88FF8E47C385640FC7D0CC1	0442FDBDE88FF8E47C385640FC7D0CC1	1.00000	0.69997	75.12477	142.00000	94.00000	0.90140	0.70797	0.57724	0.36576
9	TEST	045189AB35312C12DD16DF1033253A41	045189AB35312C12DD16DF1033253A41	1.00000	0.58990	72.36869	34.00000	66.00000	0.84303	0.61180	0.13821	0.25681
10	TRAIN	05C96477B6CC75E183B0D9AFD5925265	05C96477B6CC75E183B0D9AFD5925265	1.00000	0.73308	78.91852	21.00000	71.00000	0.91895	0.84034	0.08537	0.27626
11	TRAIN	06CF27CDCC4E54F2E6F97A87C4D1C280	06CF27CDCC4E54F2E6F97A87C4D1C280	1.00000	0.80682	80.16100	30.00000	62.00000	0.95805	0.88370	0.12195	0.24125
12	TRAIN	072CF49EFD9285A2F8B2718CE8DFAFD5	072CF49EFD9285A2F8B2718CE8DFAFD5	1.00000	0.75783	77.24422	13.00000	47.00000	0.93208	0.78192	0.05285	0.18288
13	TRAIN	083C7319AB5ED2B6F17185920AB747B0	083C7319AB5ED2B6F17185920AB747B0	1.00000	0.70943	77.24299	16.00000	75.00000	0.90641	0.78188	0.06504	0.29183
14	TRAIN	08B8ED4FC147E325869D22AA9568F5AF	08B8ED4FC147E325869D22AA9568F5AF	1.00000	0.57944	73.96440	30.00000	94.00000	0.83749	0.66748	0.12195	0.36576
15	TRAIN	0946A0775473EC8D5E33B7C75625DE1B	0946A0775473EC8D5E33B7C75625DE1B	1.00000	0.83057	80.37768	19.00000	81.00000	0.97065	0.89126	0.07724	0.31518
16	TRAIN	0AFC420793E0F6D6F1FFB8F410DA5D28	0AFC420793E0F6D6F1FFB8F410DA5D28	1.00000	0.67810	74.11942	62.00000	57.00000	0.88980	0.67289	0.25203	0.22179
17	TRAIN	0B4D2987676106558B52660F5D7ED967B	0B4D2987676106558B52660F5D7ED967B	1.00000	0.58274	66.21410	50.00000	108.00000	0.83924	0.39704	0.20325	0.42023
18	TRAIN	0BB05CAAF91A193218512769586A3968	0BB05CAAF91A193218512769586A3968	1.00000	0.68269	77.17257	30.00000	87.00000	0.89223	0.77942	0.12195	0.33852
19	TRAIN	0BB63E15DEBC84CEFEFB7C3377A47EA4	0BB63E15DEBC84CEFEFB7C3377A47EA4	1.00000	0.70702	77.37684	205.00000	228.00000	0.90513	0.78655	0.83333	0.88716

Figure 3.1: Screenshot of internal Score matrix derived from the NIST BSSR1 dataset.

3.1.1 Naming Conventions

There are a number of ways this app relies upon consistent naming conventions. We describe these instances below.

- **Original, Unedited Modalities: "_ORIGINAL"**. Original scores read in from the data files are saved in separate columns, so that modality edits or different normalization techniques can be applied without restarting the app.

- **Fusion Rules: ":"** When analyzing the performance of fusion experiments, it is easy to identify the fused score representations by collecting columns which contain ":". This means that modalities must not use ":" in the name, and that fused modalities must be identified by using ":" in their names.

3.1.2 Columns

The following describe the columns within the internal score matrix.

Train_Test

This column specifies whether a probe should belong to the TRAIN or TEST set. This ensures that when performing fusion experiments, results are comparable.

Probe_ID, Gallery_ID

These columns provide the identifier of the probe and the gallery template compared to that probe to produce the match scores.

Original Scores

Original scores are retained in the score matrix so that the user can choose a different normalization technique or edit modalities without having to restart the app.

Normalized Scores

All fusion techniques are applied to the modalities' normalized scores. Each modality contains a column of normalized scores. Normalized scores serve as the baseline modalities.

Fused Scores

There is a fused column for each fusion rule applied. These columns can be easily identified as all fused score modalities contain ":" in the name. This steps allows the separation of baseline modalities and fused modalities.

3.2 Adding Fusion Rules

There are 2 sections of code that need to be updated to add a new fusion rule: the fuse object and the GUI itself.

3.2.1 The Fuse Object

The Fuse object is defined in Fuse.py (https://github.com/melissadale/ScoreFusionApp/blob/master/App/Analytics/Fuse.py). This file can be updated with a function for your new fusion rule, keeping the following in mind:

- It is important to use only training data for any parameter setting, in order to be consistently compared to other fusion rules.

- **Use a ':' in the rule name**. Much of the app determines the fused scores by checking for a ':'. For instance, the simple sum fusion rule is titled 'SIMPLE_SUM:'.
- It is a good idea to save the important parameters of the new rule so that it may be saved as an experiment model. The following line should be included in some form a the end of your function:

```
self.models.append(TrainedModel(title=sum_title,
    train_time=t1-t0, model=None))
```

3.2.2 Updating the GUI

Currently, the Fusion panel code resides in the master styles.kv class. This is a god class and will be broken up into its own kv class in the future. In the meantime, it is possible to get to the relevant code by searching for *text: 'Fusion'*.

From here, the new rule will need to have an additional check box by adding something like:

```
CheckBox:
    id: chk_newrule
    size_hint_x: 0.2
    color: hex('#000000')

Label:
    text: 'New Fusion Rule'
    size_hint_x: 0.8
    halign: 'left'
    valign: 'middle'
    text_size: self.width, None
    color: hex('#0DB14B')
```

Next add the chk_newrule to the header of the <Main> object (located towards the top of the Styles.kv file), and to the fuse function inside of the GUI.py class.

3.3 Software UML

The following image is the layout of the Score Fusion App code development 3.2.

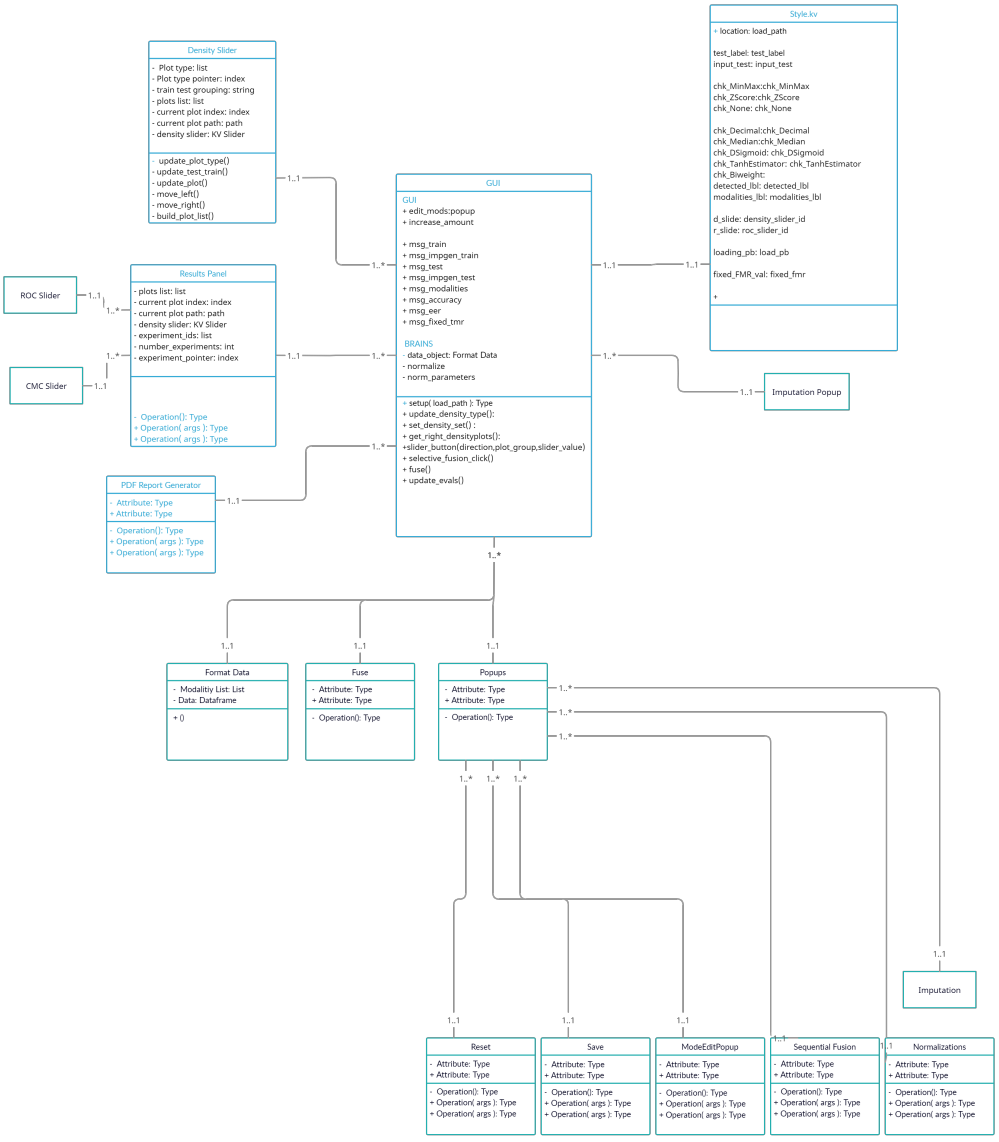


Figure 3.2: UML of Score Fusion App Design

Chapter 4

Trouble Shooting

4.1 Issue

Only test or training data is being loaded. Ensure that the word "train" or "test" is not in any of the parent directories. If either of these words appear in the path, the app will believe data is already training or testing data.

I click on a button, but nothing is happening. If nothing is happening, check that you have performed necessary previous steps. For instance, attempting to apply sequential fusion without choosing a baseline modality.

4.2 Known Issues

If running the Score Fusion App on a very large dataset or on a machine with few resources, the following issues are known to occur.

- **Image or GIF turns to black square**. This can occur when computational resources are limited, it does not impact the performance of the Score Fusion App.
- **App says Application not Responding**. This warning appears when performing a computationally expensive task, such as using KNN imputation on a very large dataset. If allowed to continue running, the score fusion app will often complete the task and return to normal.

Chapter 5

Feature Requests, Support, and Future Directions

5.1 Feature Requests and Support

Feature Requests or Bugs can be submitted at: http://www.github.com/melissadale/ScoreFusionApp/issues/new

For help or to inquire about this app, contact the author directly at: dalemeli@msu.edu

5.2 Future Directions

This project contains a number of god classes, which are actively being refactored. In addition, the author acknowledges a number of spaghetti code instances as a result of scope creep in addition to the process of learning app development with the Kivy language. A priority moving forward is to repay the technical debt that has been incurred over the course of this project.

Bibliography

[1] Arun A Ross, Karthik Nandakumar, and Anil K Jain. *Handbook of multibiometrics*. Vol. 6. Springer Science & Business Media, 2006 (page 1).