Face Analysis Technology Evaluation
Ongoing

Age Estimation And Verification (AEV)
Application Programming Interface (API)

VERSION 2.0

Kayee Hanaoka
Mei Ngan
Patrick Grother
Information Access Division
Information Technology Laboratory

April 11, 2025

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

TABLE OF CONTENTS

LIST OF TABLES

1. Face Analysis Technology Evaluation: Age Estimation And Verification (FATE AEV)

1.1. Scope

This document specifies how NIST will evaluate prototype algorithms that inspect images or video of a person's face and return an estimate of their age, or verify their age. The kind of results this activity yields can be seen in NIST Interagency Report 8525 and its revisions.

Our approach is to run an ongoing, open ended, transparent large-scale black box evaluation in which any developer can submit compiled library software to NIST and for this to be executed on images on several non-public datasets secured at NIST. The test is repeatable, fair, statistically robust, and very difficult to game. As an independent government laboratory, the cost of participation is zero. Participation procedures are on the AEV webpage.

The evaluation is intended to generically support applications including:

1. Verification that a person is above 18, 21 or other key ages, e.g. for sale of alcohol
2. Verification that a person is below a certain age, e.g. entrance to a teen chat room
3. Producing population age statistics for people visiting certain locations, e.g. movie theaters
4. Digital advertising where a display might show an age-tailored advertisement
5. Checks that a passport application photo is recent (not collected 10 years previously)
6. Age estimation for refugees, asylum seekers, and other undocumented individuals

Applications 1 and 2 are recently the subject of legislation in multiple jurisdictions. The mechanism for estimating age is often not specified in legislation. Face analysis using software is one approach, and is attractive when a photograph is available or can be captured. As succinctly noted in this infographic, age verification can also be achieved using face recognition[1] to verify the identity of an individual whose age is known from, for example, an authoritative ID document.

This evaluation is not a certification program. It does not establish minimum accuracy criteria, because different applications have different requirements.

Please direct comments and questions to frvt@nist.gov.

1.2. General Evaluation Specifications

General and common information shared between all tracks of the FRTE/FATE evaluations are documented in a General Evaluation Specifications document. This includes rules for participation, hardware and operating system environment, software requirements, reporting, and common data structures that support the APIs.

1.3. Reporting

For all algorithms that complete the evaluation

- ➔ NIST will publish on its website the name of the developer's organization, the algorithm identifiers, and the performance results with attribution to the developer.
- ➔ NIST may provide other performance results to the participating organization.
- ➔ NIST may additionally report and share results with partner government agencies and interested parties, and in standards meetings, workshops, conferences, conference papers, presentations, and technical reports.

[1] Face recognition technologies are tasked to determine who is in an image, rather than just their age; they are built on different neural network architectures trained to optimize different loss functions, and operate by comparing two photos. They are evaluated by NIST in the separate FRTE program.

Important: The developer's name will be published alongside the results.

Results will be machine generated (i.e., scripted) and will include timing, accuracy and other performance results. These will be provided alongside results from other implementations. Results will be expanded and modified as additional implementations are tested, as new image datasets become available, and as analyses are implemented. Results may be updated, extended and regenerated after an initial publication for example whenever additional implementations complete testing, or when new analyses are added.

1.4. Performance Metrics

This test will evaluate algorithmic ability to estimate the age of a person in an image or a video. NIST will measure age estimation accuracy across various imaging conditions and various demographic groups. Similarly, NIST will measure age verification accuracy. An initial set of results and analyses and visualizations can be seen in NIST Interagency Report 8525.

The results may be relevant to development of the accuracy criteria now being standardized in ISO/IEC 27566-3.

1.5. Image datasets

NIST will execute the software on an open-ended set of different image databases that are appropriate to various use cases. Primary interest is in subjects making deliberate cooperative presentations to a camera.

While NIST cannot release sample images from our operational test sets, there are many academic datasets that may be relevant for development. One such set is NIST Special Database 32 which includes two types of photos: Mugshot booking photos and smaller, lower quality, webcam photos. We will not use public sets in the AEV evaluation.

2. Implementation requirements

2.1. Library naming convention

The core library shall be named as libfrvt_ae_<***provider***>_<***sequence***>.so, with

- ➔ provider: single word, non-infringing name of the main provider. Example: acme
- ➔ sequence: a three digit decimal identifier to start at 000 and incremented by 1 every time a library is sent to NIST. Example: 007

Example core library names: libfrvt_ae_acme_000.so, libfrvt_ae_mycompany_006.so.

Important: Public results will be attributed with the provider's name and the 3-digit sequence number.

2.2. Required execution durations

The elemental functions of the implementations shall execute under the time constraints of Table 1. These time limits apply to the function call invocations defined in section 4. The median duration will be compared with the time limit meaning that the median duration should be less than that given in the Table.

The time limits apply to the processing of one image using a single core of an unloaded server-class computer equipped with Intel(R) Xeon(R) Gold 6140 CPUs running at 2.30GHz.

Table 1 – Processing time limits in milliseconds for a single 640 x 480 image of one face

Function		Age Estimation
estimateAge()	Table 3	1000 (1 core)
verifyAge()	Table 5	
estimateAgeWithReference()	Table 4	2000 (1 core)

3. Data structures and datatypes supporting the API

The data structures supporting this API are documented in the General Evaluation Specifications document, with corresponding header file named *frvt_structs.h* published at https://github.com/usnistgov/frvt.

The FATE AEV evaluation requires software to produce continuous age estimates, using a double precision datatype. This is important for applications where fractional estimates can be estimated reliably and are of interest, e.g. in neonates and very young children. We recommend providing algorithms capable of returning fractional estimates, even though some adult-focused age-verification specifications require integer estimates [ACCS].

NOTE: Humans conventionally report their age rounded down to the nearest integer - If Alice is 18 years and 11 months old, she will state her age as 18. Biological ageing will not respect such truncation, so an algorithm trained on human-reported ages may underestimate continuous age. NIST's largest evaluation sets have continuous to-the-day age metadata.

4. API Specification

FATE AEV participants shall implement the relevant C++ prototyped interfaces in Section 4.3. C++ was chosen in order to make use of some object-oriented features.

4.1. Header File

The prototypes from this document will be written to a file named **frvt_ae.h** and will be available to implementers at https://github.com/usnistgov/frvt.

4.2. Namespace

All supporting data structures will be declared in the `FRVT` namespace. All API interfaces/function calls for this track will be declared in the `FRVT_AE` namespace.

4.3. API

4.3.1. Interface

The software under test must implement the interface `Interface` by subclassing this class and implementing each method specified therein.

	C++ code fragment	Remarks
1.	`class Interface`	
2.	`{` `public:`	
3.	`virtual ReturnStatus initialize(` `const std::string &configDir) = 0;`	Supports algorithm initialization e.g. reading of DNN models from file.

4.	`virtual ReturnStatus estimateAge(`
	`const Media &face,`
	`double &age) = 0;`
		Age (in years) estimation, given either one or more contemporaneous still images, or a sequence of video frames. This function is **mandatory**.

5.	`virtual ReturnStatus estimateAgeWithReference(`
	`const Media &faceOne,`
	`const double ageOne,`
	`const Media &faceTwo,`
	`double &ageTwo) = 0;`
		Estimate the age (in years) from one or more contemporaneous still images, or a sequence of video frames of the person in faceTwo.
		faceOne and ageOne are provided as inputs for additional information of the same person in faceTwo.
		This function is **optional**.

6.	`virtual ReturnStatus verifyAge(`
	`  const Media &face,`
	`  const double ageThreshold,`
	`  double &score) = 0;`
		Return a score on how likely the face in the image, or sequence of video frames, is above the provided age threshold. This function is **optional** and will only call with the following age thresholds {13, 16, 18, 21, 25}.

7.	`static std::shared_ptr<Interface> getImplementation();`	Factory method to return a managed pointer to the `Interface` object. This function is implemented by the submitted library and must return a managed pointer to the `Interface` object.
8.	`};`	

There is one class (static) method declared in `Interface.getImplementation()` which must also be implemented by the implementation. This method returns a shared pointer to the object of the interface type, an instantiation of the implementation class. A typical implementation of this method is also shown below as an example.

	C++ code fragment	Remarks

	`#include "frvt_ae.h"`
	
	`using namespace FRVT_AE;`
	
	`NullImpl::NullImpl () { }`
	
	`NullImpl::~NullImpl () { }`
	
	`std::shared_ptr<Interface>`
	`Interface::getImplementation()`
	`{`
	`  return std::make_shared<NullImpl>();`
	`}`
	
	`// Other implemented functions`	

4.3.2. Initialization

The NIST test harness will call the initialization function in Table 2 before calling any of the quality assessment functions of this API. This function will be called BEFORE any calls to `fork()`[2] are made.

Table 2 – Initialization

Prototype	ReturnStatus initialize(const string &configDir);		Input
Description	This function initializes the implementation under test. It will be called by the NIST application before any calls to the age estimation functions of this API. The implementation under test should set all parameters. This function will be called N=1 times by the NIST application, prior to parallelizing M >= 1 calls to any other functions via `fork()`.		
Input	configDir	A read-only directory containing any developer-supplied configuration parameters or run-time data files. The name of this directory is assigned by NIST, not hardwired by the provider. The names of the files in this directory are hardwired in the implementation and are unrestricted.	
Output	none		
Return Value	See General Evaluation Specifications document for all valid return code values.		

[2] Note the copy-on-write semantics of fork() on linux: http://man7.org/linux/man-pages/man2/fork.2.html

4.3.3. Estimate age from an input media of a person

The estimateAge function takes an image and returns an estimate of the person's age.

Table 3 – Estimate age from an input media

Prototypes	ReturnStatus estimateAge(		
	const Media &face,		Input
	double &age);		Output
Description	This function estimates the age of a person from one or more still images, collected contemporaneously, or a sequence of video frames of exactly one person. An estimated age should be returned - the estimate should be fractional e.g. 18.4		
Input	face	Input media (stills or video frames) of one person.	
Output	age	Estimated age of the person in fractional years - see discussion in section 3.	
Return Value	See General Evaluation Specifications document for all valid return code values. This function **must** be implemented.		

4.3.4. Estimate age from an input media given ground truth from separate media

The estimateAgeWithReference function takes a face image for which the age is known, and a second face image for which the age must be estimated. This could improve accuracy if there exist people who "look young for their age", for example.

Table 4 – Estimate age given ground truth from separate media

Prototypes	ReturnStatus estimateAgeWithReference(		
	const Media &faceOne,		Input
	const double ageOne,		Input
	const Media &faceTwo,		Input
	double &ageTwo);		Output
Description	This function estimates the age (in years) of the person in faceTwo. This function allows an implementation to either estimate age as in section 4.3.3, or to exploit the additional information inherent in the faceOne imagery. Developers should not assume that faceTwo is collected at a later date than faceOne.		
Input	faceOne	Input media (stills or video frames) of one person.	
Input	ageOne	Actual age of the person in faceOne at the time it was taken e.g. 18.4 years.	
Input	faceTwo	Input media (stills or video frames) of the same person in faceOne.	
Output	ageTwo	A value indicating the estimated age of the face in faceTwo. The estimate should be fractional - see discussion in section 3.	
Return Value	See General Evaluation Specifications document for all valid return code values. If this function is **not implemented**, the return code should be set to `ReturnCode::NotImplemented`.		

4.3.5. Age Compliance

This verifyAge function takes a face image and an age threshold, then returns a score indicating how likely the person is above that age threshold.

Table 5 – Verification that subject in input media is above an age threshold

Prototypes	ReturnStatus verifyAge(	
	const Media &face,	Input
	const double ageThreshold,	Input
	double &score);	Output
Description	This function returns a score, with higher values indicating a greater likelihood that the person's age exceeds the specified threshold. This function prototype allows an implementation to invoke specialized processing for certain age groups. We anticipate calling this function with ageThreshold values in {13, 16, 18, 21, and 25}.	
Input	face	Input media (stills or video frames) of one person.
Input	ageThreshold	Input age of interest.
Output	score	A real-valued score whose range is not limited by this document, with larger values indicating that it is more likely that the face in the media is above the age threshold.
Return Value	See General Evaluation Specifications document for all valid return code values. If this function is **not implemented**, the return code should be set to `ReturnCode::NotImplemented`.	