Dlib FaceLandmark Detector 1.2.1

WebGL support
iOS & **Android** support
Windows10 UWP support
Win & **Mac** & **Linux** Standalone support
Support for preview in the **Editor**
Works with **Unity Cloud Build**

System Requirements
Build Win Standalone & Preview Editor : Windows7 or later
Build Mac Standalone & Preview Editor : OSX 10.9 or later

DlibFaceLandmarkDetector can **Object Detection** and **Shape Prediction** using Dlib19.7 C++ Library.

Features:

- You can detect **frontal human faces and face landmark (68 points)** in **Texture2D**, **WebCamTexture** and **Image byte array**. In addition, you can detect a different objects by changing trained data file.
- **Object Detector** is made using the now classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme. You can train your own detector in addition to human faces detector. If you want to train your own detector, please refer to this page.
- **Shape Predictor** is created by using dlib's implementation of the paper (One Millisecond Face Alignment with an Ensemble of Regression Trees by Vahid Kazemi and Josephine Sullivan, CVPR 2014). You can train your own models in addition to human face landmark model using dlib's machine learning tools. If you want to train your own models, please refer to this page.
- Advanced examples using "**OpenCV for Unity**" are Included.(The execution of this examples are required "OpenCV for Unity".)
- PlayMakerActions for DlibFaceLandmarkDetector is available.

Basic Examples:

- Texture2D Example
- WebCamTexture Example
- Cat Detection Example

Advanced Examples (require OpenCV for Unity):

- Texture2DToMat Example
- WebCamTextureToMatHelper Example
- AR Head WebCamTexture Example
- AR Head VideoCapture Example
- Frame Optimization Example
- NoiseFilter WebCamTexture Example
- NoiseFilter VideoCapture Example

Official Site | ExampleCode | Android Demo WebGL Demo | Setup Tutorial & Demo Video | Forum | API Reference

Version changes

1.2.1 [Common]Updated to WebCamTextureToMatHelper.cs v1.0.8. [Common]Updated to LowPassPointsFilter v1.0.1. Updated to KFPointsFilter v1.0.2. Updated to OFPointsFilter v1.0.2. [Common] Added updateMipmaps and makeNoLongerReadable flag to DrawDetectResult () and DrawDetectLandmarkResult() method. [Common]Fixed Utils.getFilePathAsync() method.(Changed #if UNITY_2017 && UNITY_2017_1_OR_NEWER to #if UNITY_2017_1_OR_NEWER.)

1.2.0 [Common]Updated to WebCamTextureToMatHelper.cs v1.0.7. [Common]Fixed WebCamTextureExample and OpenCVForUnityUtils. [Common]Added NoiseFilterVideoCaptureExample and NoiseFilterWebCamTextureExample. [Commo]Added useLowPassFilter option to ARHeadVideoCaptureExample and ARHeadWebCamTextureExample. [Common]Added throwException flag to Utils.setDebugMode() method. [Common]Added drawIndexNumbers flag to

DrawFaceLandmark() method.

1.1.9 [Android]Added arm64-v8a Architecture.

1.1.8 [Common]Updated WebCamTextureExample.(support Portrait ScreenOrientation) [Common]Updated to WebCamTextureToMatHelper.cs v1.0.4.

1.1.7 [Common]Updated “human_face_68_sp.dat” and “human_face_68_sp_for_mobile.dat”.

1.1.6 [Common]Updated to dlib19.7. [Common]Updated to WebCamTextureToMatHelper.cs v1.0.3. [Common]Updated “human_face_68_sp.dat” and “human_face_68_sp_for_mobile.dat”.

1.1.5 [Common]Switched to the shape predictor file trained using new datasets.

1.1.4 [Common]Updated WebCamTextureToMatHelper.cs v1.0.2 [Common]Improved Utils.getFilePathAsync().

1.1.3 [Common]Fixed to improve the pose estimation performance. [Common] Changed DetectLandmarkArray (int left, int top, int width, int height) to DetectLandmarkArray (double left, double top, double width, double height). [WebGL]Fixed Utils.getFilePathAsync() method.

1.1.2 [Common]Updated WebCamTextureToMatHelper.cs and OptimizationWebCamTextureToMatHelper.cs(Changed several method names.). [Common]Changed the Example name.

1.1.1 [Common]Improved Utils.getFilePath() and Utils.getFilePathAsync().

1.1.0 [Win][Mac][Linux][UWP]Added the native plugin file enabled SSE4 or AVX compiler option.

1.0.9 [WebGL]Added WebGL Plugin for Unity5.6.

1.0.8 [Common]Changed the name of asset project.("Sample" to "Example") [Common]Fixed VideoCaptureARExample and WebCamTextureARExample.

1.0.7 [Common]Fixed WebCamTextureToMatHelper.cs.(flipVertical and flipHorizontal flag)

1.0.6 [Common]Fixed OpenCVForUnityMenuItem.cs.(No valid name for platform: 11 Error) [Common]Added OptimizationWebCamTextureToMatHelper.cs

1.0.5 [Common]Fixed WebCamTextureToMatHelper class. [Common]Added Utils.getVersion(). [Common]Fixed Utils.getFilePathAsync().

1.0.4 [Common]Updated shape_predictor_68_face_landmarks_for_mobile.dat.

1.0.3 [WebGL]Added WebGL(beta) support.(Unity5.3 or later) [Common]Fixed missing script error.(WebCamTextureToMatHelper.cs) [Common]Added shape_predictor_68_face_landmarks_for_mobile.dat.

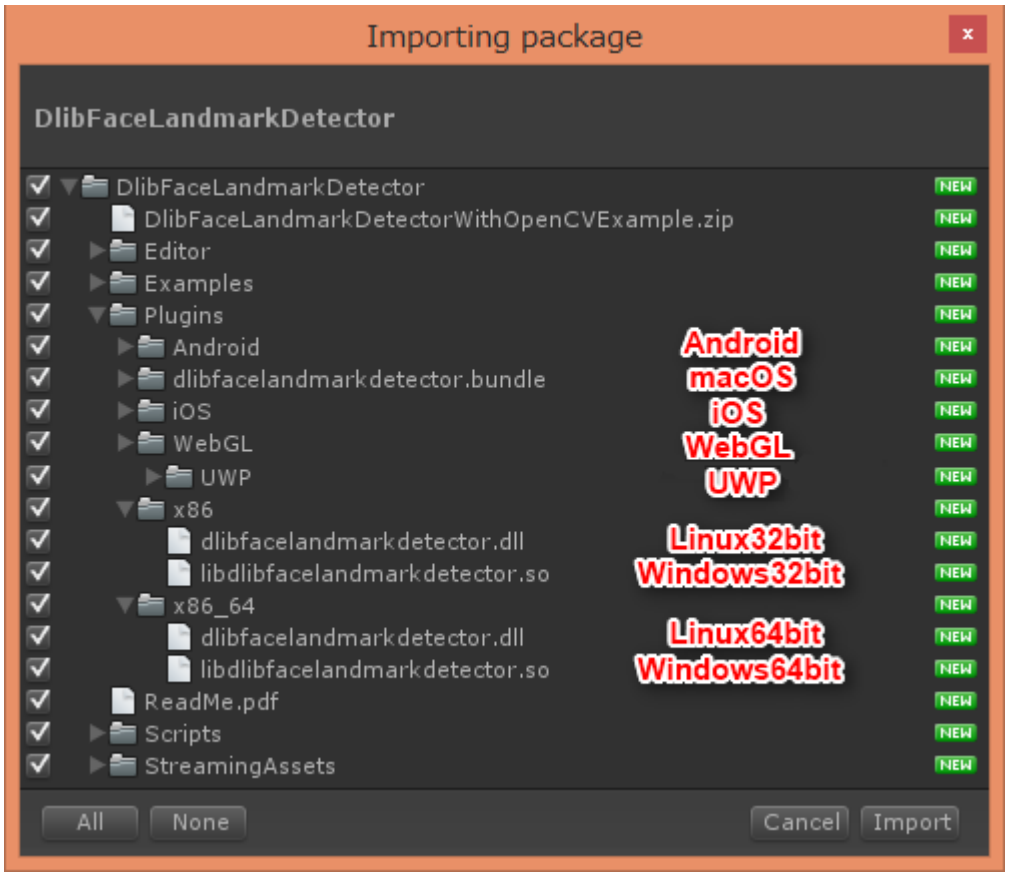
1.0.2 [Common]Improved WebCamTextureHelper class.

1.0.1 [Common]Added OptimizationSample. [Common]Added DetectRectDetection() method.

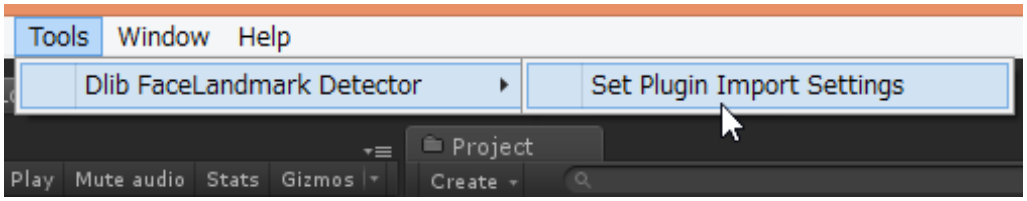
1.0.0 Initial version

Quick setup procedure to run the example scenes (Setup Tutorial Video)

1. Import the DlibFaceLandmarkDetector.package. You do not need to import plug-in files for platforms not supported by your project.



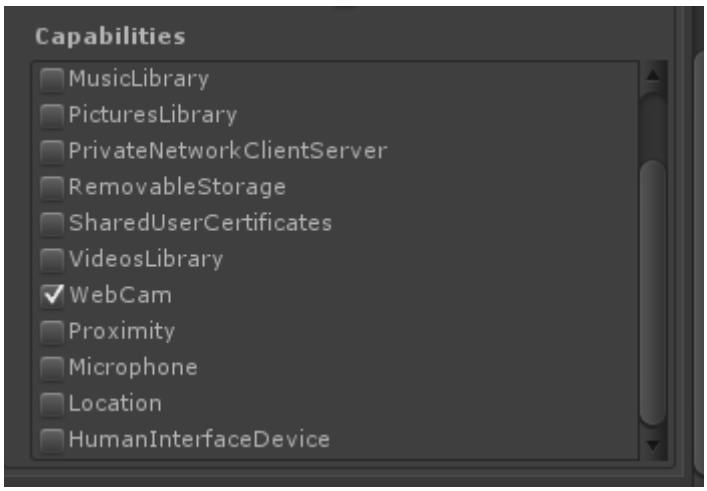
2. Select MenuItem[Tools/Dlib FaceLandmark Detector/Set Plugin Import Settings].



3. Move the “DlibFaceLandmarkDetector/StreamingAssets/” folder to the “Assets/” folder.

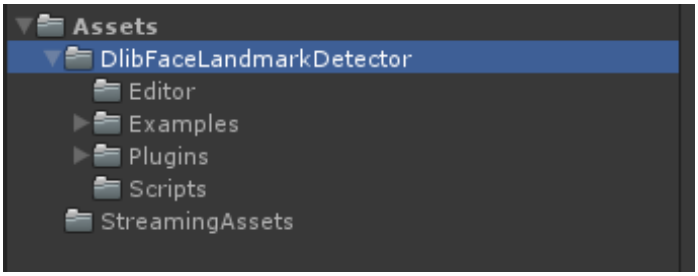
4. Add all of the "***.unity" in the "DlibFaceLandmarkDetector" folder to [Build Settings] – [Scene In Build].
5. **[iOS]** Set [PlayerSettings]-[Other Settings]-[Configuration]-[Camera Usage Description].

6. **[WindowsStoreApps8.1 & WindowsPhone8.1 & Windows10 UWP]** If use webCamTextue class, Please choose "WebCam" in [PlayerSettings]-[PublishingSettings]-[Capabilities].



7. **[Linux]** Additional Setting is required to run on the editor. http://forum.unity3d.com/threads/native-plugin-in-editor-steam-specifically.384970/

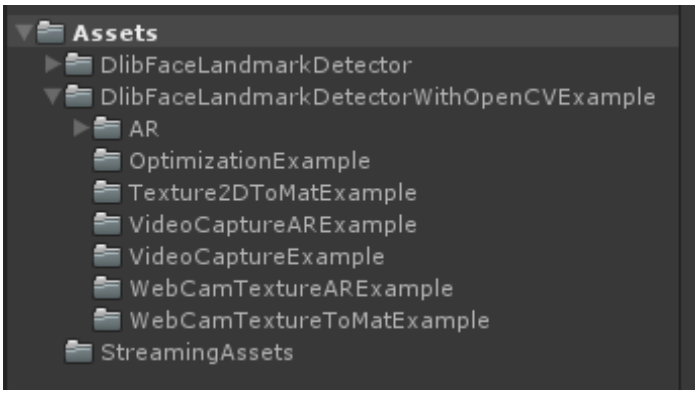
Screenshot after the setup



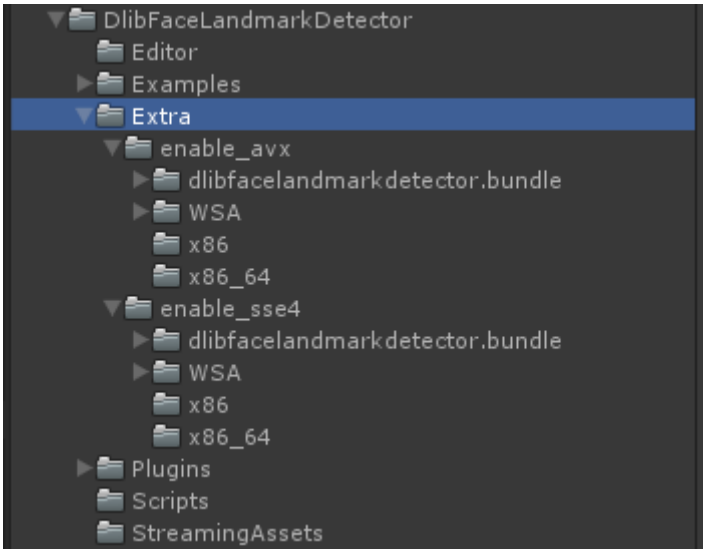
Quick setup procedure to run the Advanced examples using "OpenCV for Unity" scene

1. Import and Setup the "OpenCV for Unity".
2. Import the DlibFaceLandmarkDetectorWithOpenCVExample.unitypackage.
3. Add all of the "***.unity" in the "DlibFaceLandmarkDetectorWithOpenCVExample" folder to [Build Settings] – [Scene In Build].

Screenshot after the setup



If you want to use the native plugin file enabled SSE4 or AVX compiler option, please replace the native plugin file in the "DlibFaceLandmarkDetector/Plugin/" folder with the "DlibFaceLandmarkDetector/Extra/" folder. And, select MenuItem[Tools/Dlib FaceLandmark Detector/Set Plugin Import Settings].



Q & A

Q1.

"DllNotFoundException: dlibfacelandmarkdetector" is displayed on the console when run the example scene.

A1.

Plugin does not seem to be loaded correctly. Please check the setup procedure.

Q2.

"Level 'Texture2DExample' (-1) could not be loaded because it has not been added to the build settings." is displayed on the console when run the example scene.

A2.

Please Add all of the "***.unity" in the "DlibFaceLandmarkDetector" folder to [Build Settings] – [Scene In Build].

Q3.

In Texture2DExample, red rectangle is not displayed around a face.

A3.

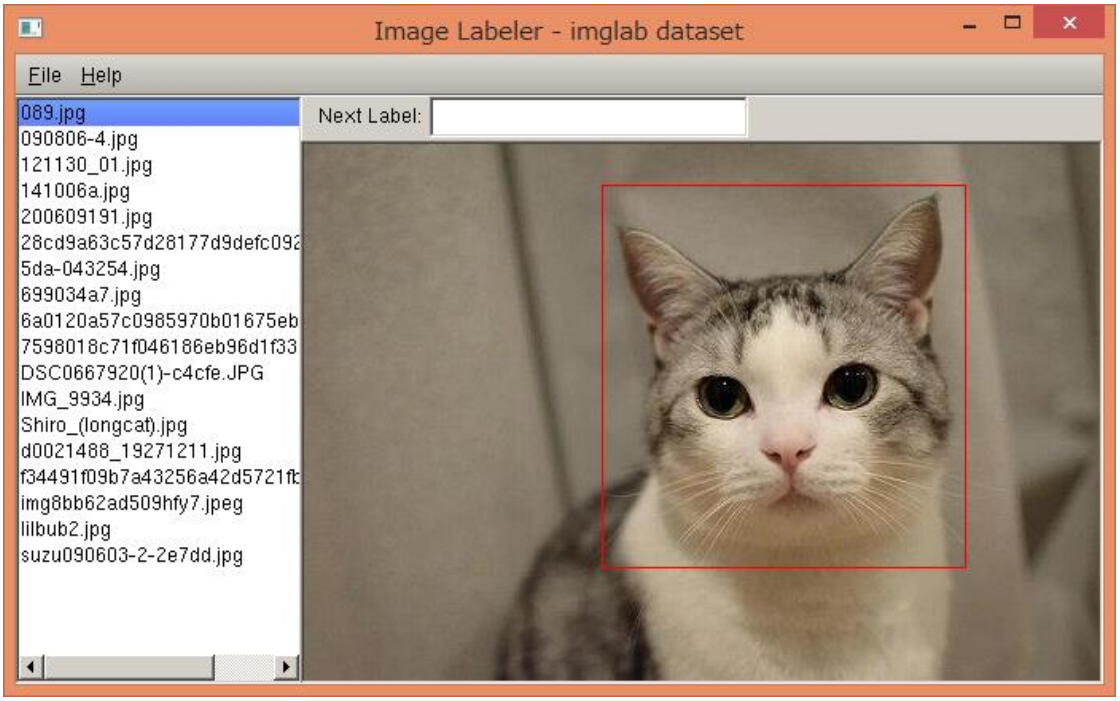
Please move the “DlibFaceLandmarkDetector/StreamingAssets/” folder to the “Assets/” folder.

Q4.

How can I train object detector?

A4.

Please refer to this page.

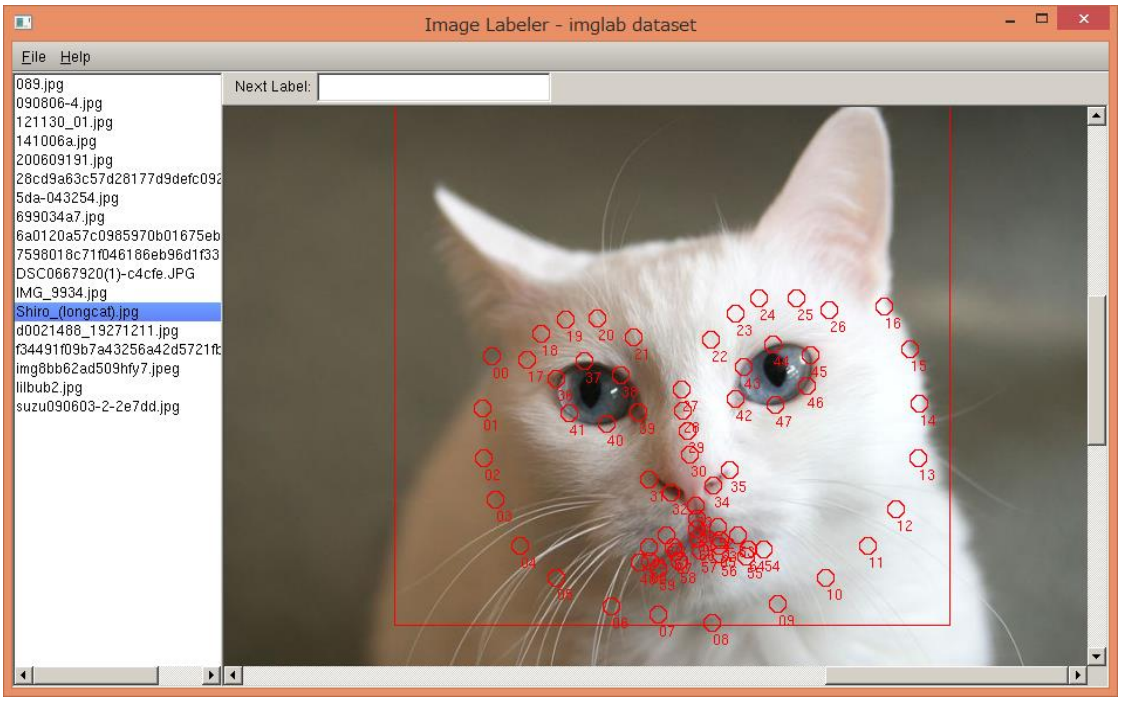


Q4.

How can I train shape predictor?

A4.

Please refer to this page.



Q5.

The size of the “human_face_68_sp.dat” is too large.

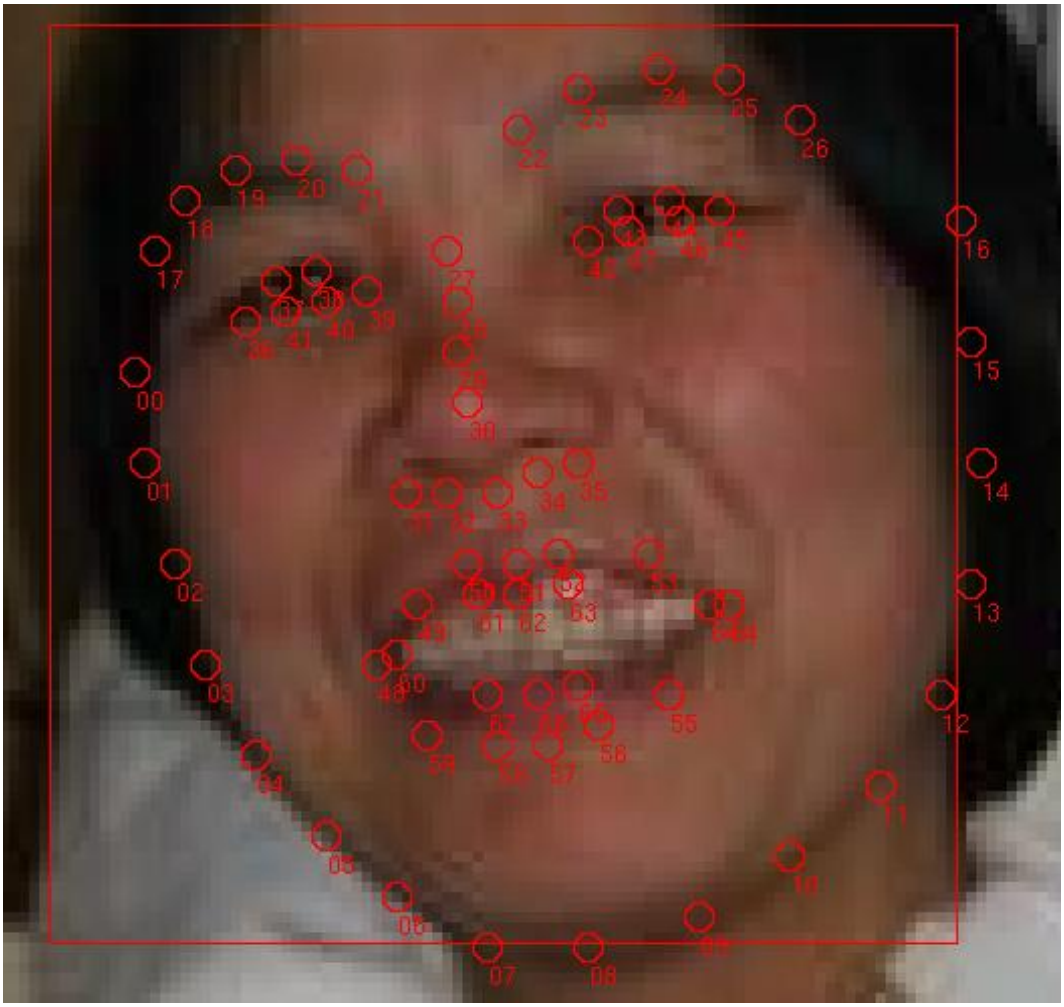
A5.

Please use the “human_face_68_sp_for_mobile.dat”. (the “human_face_68_sp_for_mobile.dat” is less accurate than the “human_face_68_sp.dat”, but it is smaller size.)

Q6.

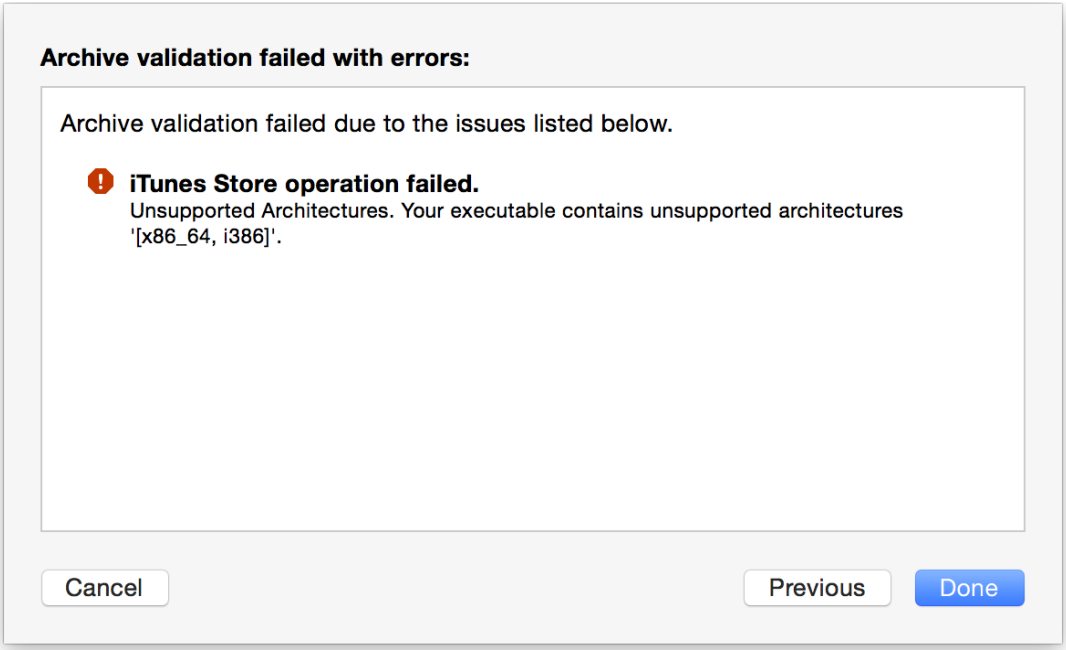
The index of face landmark points that can be obtained using the “human_face_68_sp.dat”.

A6.



Q7.

[iOS]Submit to App Store issues: Unsupported Architecture x86, i386“Unsupported Architecture. Your executable contains unsupported architecture ‘[x86_64, i386]’.”



A7.

"The problem is that the Buy framework contains a build for both the simulator (x86_64) and the actual devices (ARM).

Of course, you aren't allowed to submit to the App Store a binary for an unsupported architecture, so the solution is to "manually" remove the unneeded architectures from the final binary, before submitting it."

http://ioscake.com/submit-to-app-store-issues-unsupported-architecture-x86.html

http://ikennd.ac/blog/2015/02/stripping-unwanted-architectures-from-dynamic-libraries-in-xcode/