



# Integrate Arm NN into an Android app

Version 21.11

## Tutorial

### Non-Confidential

Copyright © 2021 Arm Limited (or its affiliates).  
All rights reserved.

### Issue 01

102744\_2111\_01\_en



# Integrate Arm NN into an Android app

## Tutorial

Copyright © 2021 Arm Limited (or its affiliates). All rights reserved.

## Release information

### Document history

Issue	Date	Confidentiality	Change
2111-01	26 November 2021	Non-Confidential	First release for 21.11

## Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly

or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2021 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

## Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

## Product Status

The information in this document is Final, that is for a developed product.

## Web address

[developer.arm.com](https://developer.arm.com)

## Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email [terms@arm.com](mailto:terms@arm.com).

# Contents

<b>1 Introduction.....</b>	<b>6</b>
1.1 Conventions.....	6
1.2 Additional reading.....	7
1.3 Feedback.....	7
1.4 Other information.....	8
<b>2 Overview.....</b>	<b>9</b>
<b>3 Integrate Arm NN in Android app.....</b>	<b>10</b>
<b>4 Tips and Tricks.....</b>	<b>18</b>
<b>5 Related information.....</b>	<b>19</b>
<b>A Revisions.....</b>	<b>20</b>

# 1 Introduction

## 1.1 Conventions

The following subsections describe conventions used in Arm documents.




### Glossary




The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm® Glossary for more information: [developer.arm.com/glossary](https://developer.arm.com/glossary).

### Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
<i>italic</i>	Introduces special terminology, denotes cross-references, and citations.
<b>bold</b>	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<i>monospace italic</i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
<b>monospace bold</b>	Denotes language keywords when used outside example code.
monospace <u>underline</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;</pre>
<b>SMALL CAPITALS</b>	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>Arm Glossary</i> . For example, <b>IMPLEMENTATION DEFINED</b> , <b>IMPLEMENTATION SPECIFIC</b> , <b>UNKNOWN</b> , and <b>UNPREDICTABLE</b> .
 Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
 Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
 Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.

Convention	Use
 Note	This represents an important piece of information that needs your attention.
 Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
 Remember	This is a reminder of something important that relates to the information you are reading.

## 1.2 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

**Table 1-2: Arm publications**

Document Name	Document ID	Licensee only
None	-	-

## 1.3 Feedback

Arm welcomes feedback on this product and its documentation.

### Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

### Feedback on content

If you have comments on content then send an e-mail to [errata@arm.com](mailto:errata@arm.com). Give:

- The title Integrate Arm NN into an Android app Tutorial.
- The number 102744\_2111\_01\_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.



Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

---

## 1.4 Other information

See the Arm website for other relevant information.

- [Arm® Developer](#).
- [Arm® Documentation](#).
- [Technical Support](#).
- [Arm® Glossary](#).



## 2 Overview

This guide provides instructions on how to use the Arm NN Tflite Delegate in an Android app using Android Studio.

The guide shows you how easy it is to integrate Arm NN into an existing app that uses the Tflite Interpreter.

### Before you begin

This guide uses Android Studio to modify an existing Android app. Therefore, you must have Android Studio installed. You can find installation instructions on the [Android-Webpage](#).

As a basis this guide uses the [Image-Segmentation example applications](#) from the [Tensorflow-Examples](#) repository. Do not forget to clone the [Tensorflow-Examples](#) repository onto your machine using git.

To try out the example yourself, you must have an Android device with a Cortex-A CPU or Mali GPU. Otherwise, you will not be able to benefit from the hardware acceleration that Arm NN offers. If you are not sure if your device has a matching CPU or GPU, you can use the utility functions in the [Integrate Arm NN in Android app](#) section to test if Arm NN is supported on your device.

## 3 Integrate Arm NN in Android app

This section shows you how to change the Image-Segmentation example to use Arm NN to run the neural network in the app.

### Introduction

For an introduction to what the Image-Segmentation app does and what its features are, read the read-me file in the examples directory `examples/lite/examples/image_segmentation/android/README.md`.

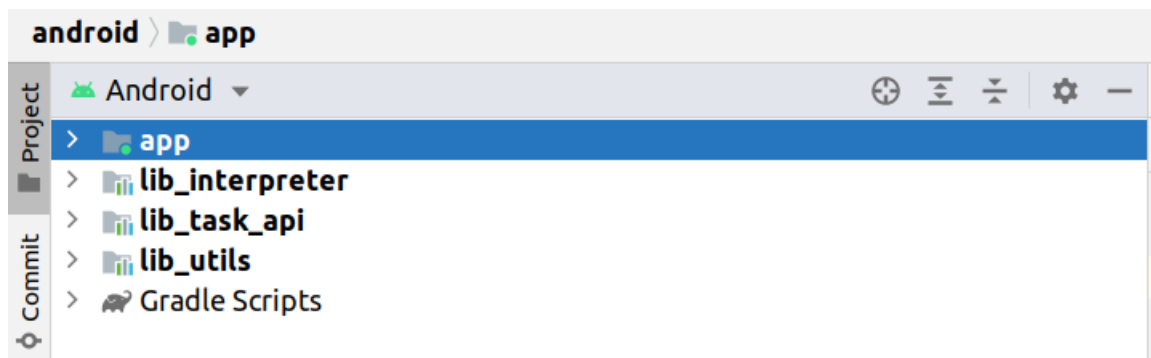
Arm NN provides an Android Library (AAR) that gives you access to the Arm NN Tflite Delegate. This library can be added to any Android app as a dependency. Once you have created an Arm NN Delegate, you can pass it on to the Tflite interpreter which uses the Arm NN delegate to execute models. Before you try that out, you must load the example app into Android Studio and test the original app. The procedure is described in the following section.

### Test the original app

Open the image-segmentation example from the Tensorflow-Examples repository with Android Studio `File > Open > examples/lite/examples/image_segmentation/android`.

The initial setup might take a moment but once that is done you will see the following structure in your Project view.

**Figure 3-1: Project structure of the Image-Segmentation app**



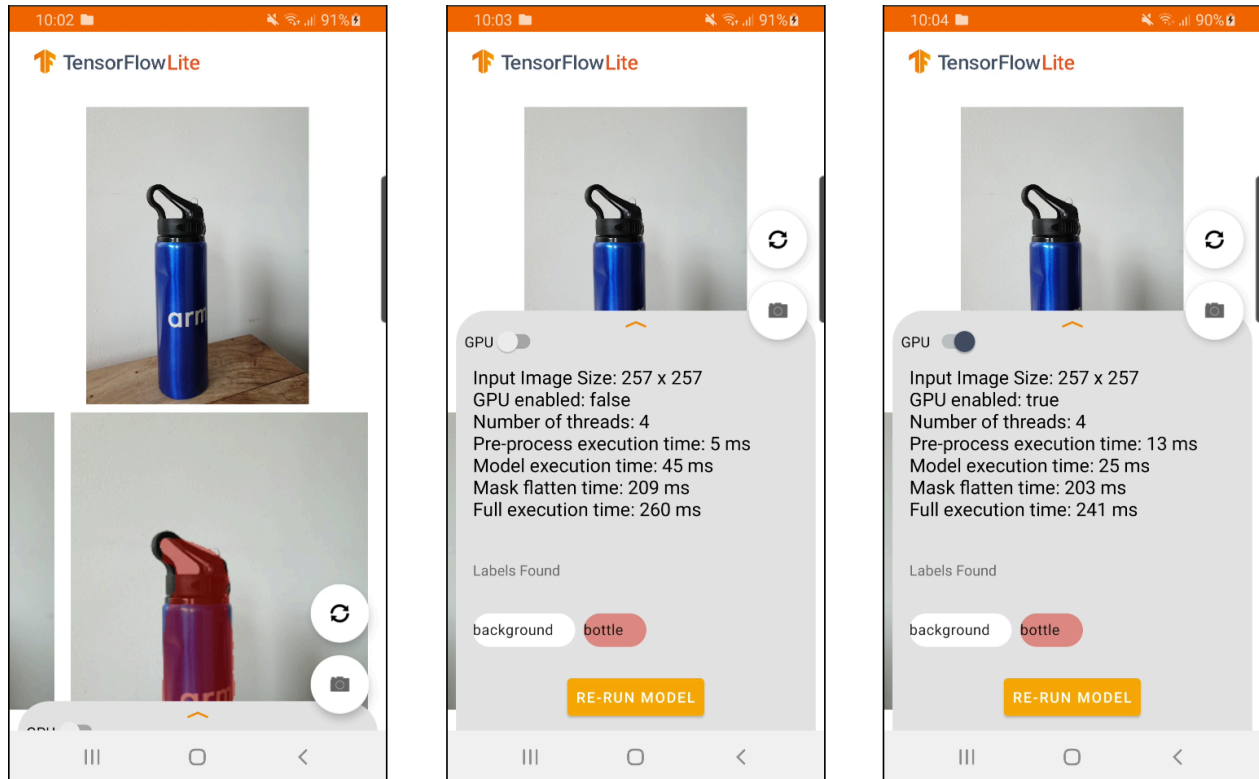
The app offers two different ways to do image segmentation, one is using the Task-API and the other is using the Tflite-Interpreter. This guide changes the interpreter variant of the app to use Arm NN. To change to the interpreter build variant go to `Build > Select Build Variant` and select `interpreterDebug`.

Once the correct build variant is selected, it is time for a test run. Connect a phone to your machine. The device must be in developer mode to be used in Android Studio. You can find more instructions on how to run apps on hardware devices [here](#).

The following figure shows the app having run correctly. In the following figure:

1. The left image shows a screen-shot of the app just after an image has been taken. The area in which the bottle was detected is marked.
2. The middle image shows execution times using the CPU of the phone.
3. The right image shows execution times when enabling the GPU using the switch button.

**Figure 3-2: Test run of the original Image-Segmentation app.**



You can use the switch button `GPU` to enable the Tensorflow Gpu-Delegate which runs the underlying neural network on the GPU. Pay attention to the reported model execution times for a later comparison.

If the app runs correctly, you can proceed to the next step.

## Modify the app

The steps in this section cover modifying the app to use Arm NN. The goal is to change the switch button to enable Arm NN instead of the GPU.

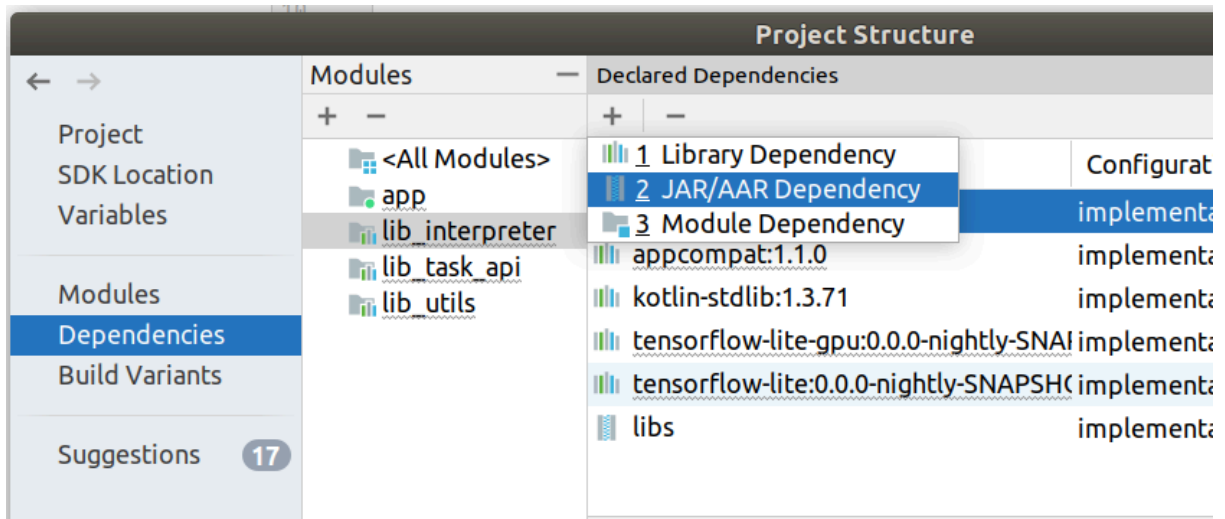
## Procedure

1. Download the Arm NN Android Library from our [GitHub release page](#). It is uploaded as an asset to the latest release.

2. Add the Arm NN module as a dependency to the `lib_interpreter` module.

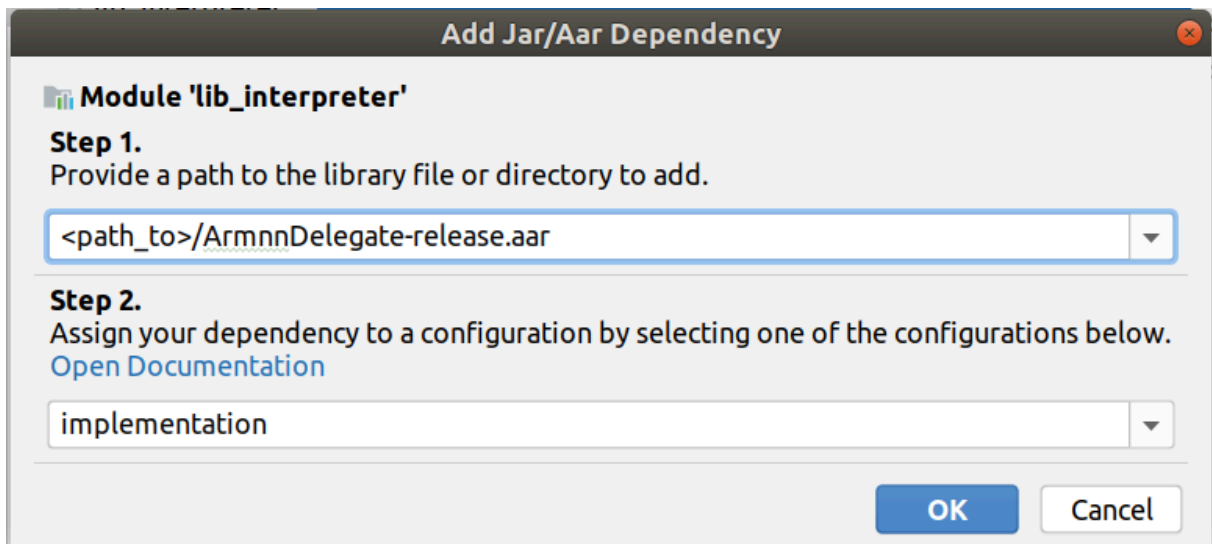
Open the Project Structure view `File > Project Structure`. Choose `Dependencies` then the `lib_interpreter` module and after that click the `+` button to add a `JAR/AAR` Dependency.

**Figure 3-3: Add the Arm NN Library as dependency**



In the next dialog, add the directory path to the Arm NN Android Library that you have downloaded in the first step.

**Figure 3-4: Add the Arm NN Library as dependency**



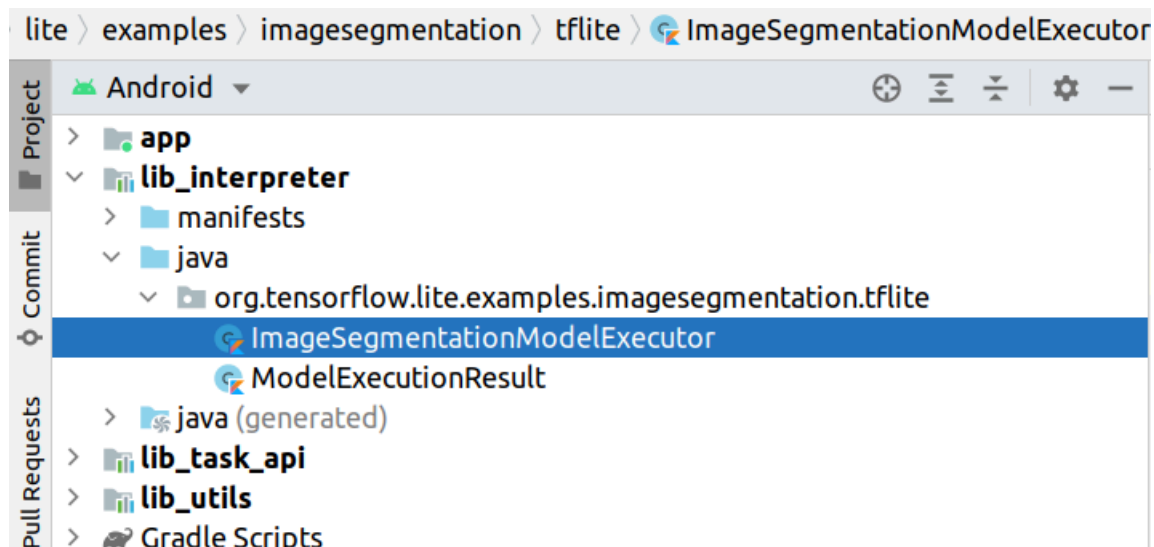
### 3. Modify the `lib_interpreter` module.

First we update the `kotlin-gradle-plugin` version to make the example compatible with the Arm NN module. To do so, navigate to the `image-segmentation/android/build.gradle` file and change the line marked with - to the line marked with +:

```
@@ -6,11 +6,11 @@ buildscript {
    mavenCentral()
}
dependencies {
    classpath 'com.android.tools.build:gradle:4.0.0'
    classpath 'de.undercouch:gradle-download-task:4.0.2'
-   classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.3.71'
+   classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.31'
    // NOTE: Do not place your application dependencies here; they belong
    // in the individual module build.gradle files
}
}
```

In the next step, the `lib_interpreter` module is changed to use Arm NN when the switch button is enabled. This follows the same procedure as the Tflite GPU delegate. Open the `ImageSegmentationModelExecutor.kt` file in the `lib_interpreter` module.

**Figure 3-5: Open ImageSementationModelExecutor.kt**



And then apply the following changes:

- Change the Boolean variable `useGPU` to `useArmNN`.
- Initialize a variable to hold an instance of the `ArmnnDelegate`.
- In the `getInterpreter` function, rename the parameter `useGPU` to `useArmNN`.
- When the Tflite interpreter is created, create an Arm NN delegate if requested. Additional options can be given to the `ArmnnDelegate` in the form of two arrays of strings. The first contains the name of the options and the second contains the value the options should be changed to. In this example, logging is turned on and set to `info`. Arm NN supports multiple backends. To run a model on Arm Cortex-A CPUs, set the backend option to

`CpuAcc`. However, if you want to run the model on Arm Mali GPUs use `GpuAcc`. There are multiple options for optimization purposes that can be configured to increase the execution speed. You can find more information on these in the Arm NN module description or in our [documentation on GitHub](#).

- Before you attempt to execute a model with `CpuAcc` or `GpuAcc`, you must confirm that the device supports that backend. This can be done with the `ArmnnUtils` functions supplied with the library.
- Because the TfLite's GPU delegate is usually faster than the CPU, we change the app to use the GPU delegate by default. That means if the switch is turned off, the GPU delegate is used and if it is turned on the Arm NN delegate is used.
- The last code change in this file is to destroy the Arm NN delegate when the `close()` function is called.

The changes that were described can be seen in the following code:

```
@@ -20,10 +20,12 @@ import android.content.Context
import android.graphics.Bitmap
import android.graphics.Color
import android.os.SystemClock
import androidx.core.graphics.ColorUtils
import android.util.Log
+import com.arm.armnn.delegate.ArmnnDelegate
+import com.arm.armnn.delegate.ArmnnUtils
import java.io.FileInputStream
import java.io.IOException
import java.nio.ByteBuffer
import java.nio.ByteOrder
import java.nio.MappedByteBuffer
@@ -45,13 +47,14 @@ import org.tensorflow.lite.gpu.GpuDelegate
 * 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike',
 * 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tv'
 */
class ImageSegmentationModelExecutor(
    context: Context,
-    private var useGPU: Boolean = false
+    private var useArmNN: Boolean = false
) {
    private var gpuDelegate: GpuDelegate? = null
+    private var armnnDelegate: ArmnnDelegate? = null

    private val segmentationMasks: ByteBuffer
    private val interpreter: Interpreter

    private var fullTimeExecutionTime = 0L
@@ -61,11 +64,11 @@ class ImageSegmentationModelExecutor(

    private var numberThreads = 4

    init {

-        interpreter = getInterpreter(context, imageSegmentationModel, useGPU)
+        interpreter = getInterpreter(context, imageSegmentationModel, useArmNN)
        segmentationMasks = ByteBuffer.allocateDirect(1 * imageSize * imageSize *
        NUM_CLASSES * 4)
        segmentationMasks.order(ByteOrder.nativeOrder())
    }

    fun execute(data: Bitmap): ModelExecutionResult {
@@ -147,28 +150,41 @@ class ImageSegmentationModelExecutor(

    @Throws(IOException::class)
    private fun getInterpreter(
```

```

        context: Context,
        modelName: String,
-       useGpu: Boolean = false
+       useArmNN: Boolean = false
    ): Interpreter {
        val tfliteOptions = Interpreter.Options()
        tfliteOptions.setNumThreads(numberThreads)

        gpuDelegate = null
-       if (useGpu) {
+       armnnDelegate = null
+       if (useArmNN) {
+           if (ArmnnUtils.isGpuAccBackendSupportedOnThisDevice()) {
+               var optionKeys = arrayOf("logging-severity", "backends", "enable-fast-
math", "reduce-fp32-to-fp16", "memory-import")
+               var optionValues = arrayOf("info", "GpuAcc",
"true", "true", "true")
+               armnnDelegate = ArmnnDelegate(optionKeys, optionValues)
+               tfliteOptions.addDelegate(armnnDelegate)
+           }
+           else {
+               Log.w(TAG, "The Arm NN GpuAcc backend is not supported on this device.")
+           }
+       }
+       // the gpu delegate is enabled by default
+       else {
            gpuDelegate = GpuDelegate()
            tfliteOptions.addDelegate(gpuDelegate)
        }

        return Interpreter(loadModelFile(context, modelName), tfliteOptions)
    }

    private fun formatExecutionLog(): String {
        val sb = StringBuilder()
        sb.append("Input Image Size: $imageSize x $imageSize\n")
-       sb.append("GPU enabled: $useGPU\n")
+       sb.append("ArmNN enabled: $useArmNN\n")
        sb.append("Number of threads: $numberThreads\n")
        sb.append("Pre-process execution time: $preprocessTime ms\n")
        sb.append("Model execution time: $imageSegmentationTime ms\n")
        sb.append("Mask flatten time: $maskFlatteningTime ms\n")
        sb.append("Full execution time: $fullTimeExecutionTime ms\n")
@@ -178,10 +194,13 @@ class ImageSegmentationModelExecutor(
    fun close() {
        interpreter.close()
        if (gpuDelegate != null) {
            gpuDelegate!!.close()
        }
+       if (armnnDelegate != null) {
+           armnnDelegate!!.close()
+       }
    }

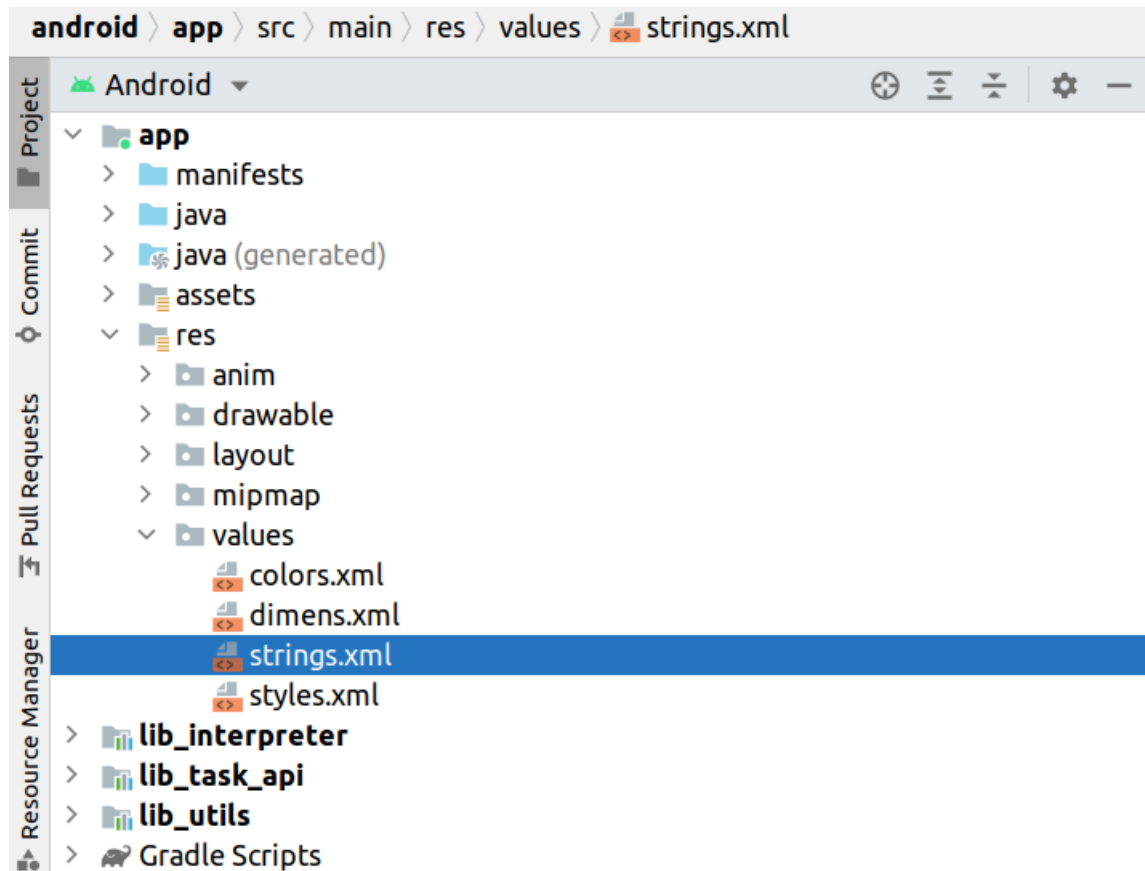
    private fun convertByteBufferMaskToBitmap(
        inputBuffer: ByteBuffer,

```

```
imageWidth: Int,
```

- To make it clearer which delegate is being used, we can change the label of the switch button from GPU to ArmNN. This can be done with the following change in `image-segmentation/android/app/src/main/res/values/strings.xml`

**Figure 3-6: Open string.xml**



```
@@ -1,7 +1,7 @@
<resources>
    <string name="tfe_is_app_name" translatable="false">TFL Image Segmenta\
tion</string>
-    <string name="tfe_is_gpu" translatable="false">GPU</string>
+    <string name="tfe_is_gpu" translatable="false">ArmNN</string>
    <string name="tfe_is_re_run_model" translatable="false">Re-run mod\
el</string>
    <string name="tfe_is_labels_found" translatable="false">Labels Found</
string>
    <string name="tfe_is_no_labels_found" translatable="false">No Labels Found</
string>
</resources>
```

## Test Arm NN

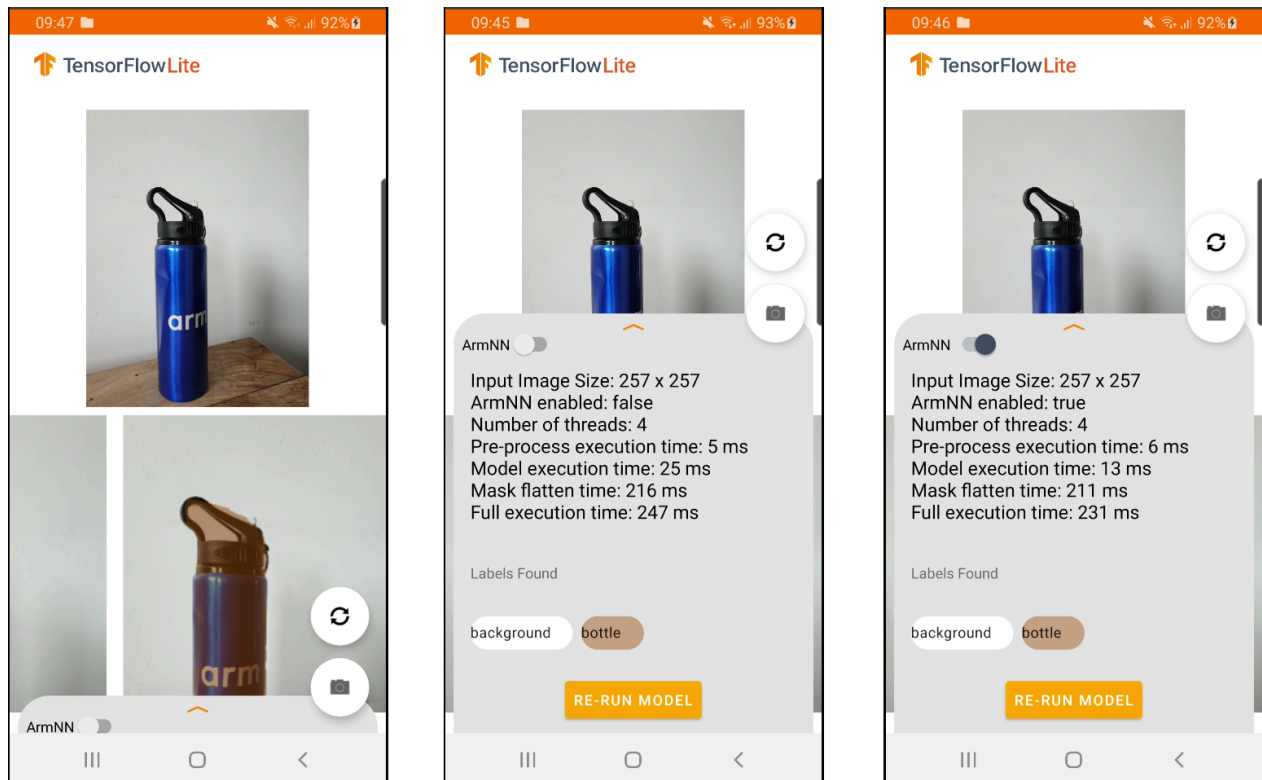
Now that all changes to the app have been made, we can try out the app.



As you can see in the following figure, the performance has improved significantly when using the Arm NN delegate. In the following figure:

1. The left image shows a screen-shot of the app just after an image has been taken. At this stage, the app uses the TFLite GPU delegate.
2. The middle image shows the performance figures with the TFLite GPU delegate.
3. The right image shows the performance using the Arm NN delegate.

**Figure 3-7: Test run of the Image-Segmentation app with Arm NN.**



When you try this out yourself, you will notice quite a bit of noise in the model execution time regardless of which backend is applied. When taking the images, we have tried to represent the average execution time we have been seeing on our test device.

These performance numbers are estimates. Even though Arm NN usually turns out to be faster, it cannot be generalized that Arm NN is always faster than the TFLite GPU delegate. The resulting performance depends on the model and the operations it contains. Models can also be optimized to run on specific hardware. The hardware in the device also plays a significant role.

## 4 Tips and Tricks

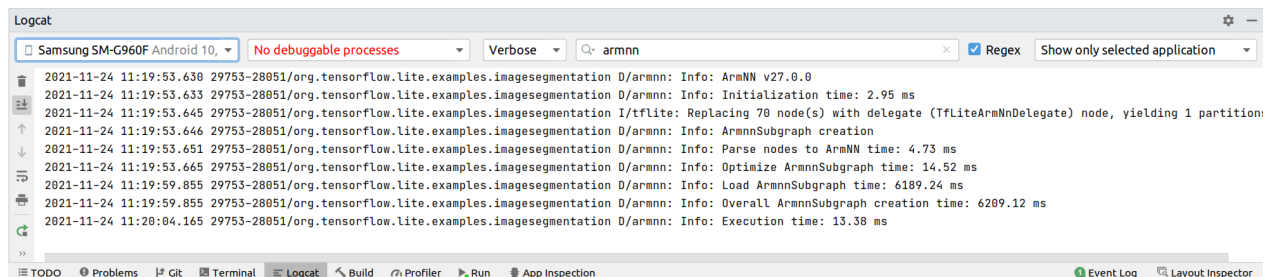
This section contains tips to help you if you have trouble with the Arm NN Android Library.

### Enable logging

If you cannot verify that Arm NN is running after you have integrated the Arm NN delegate, look at the logging output in logcat. You can view the logging output in Android Studio.

Apply a filter `armnn` to the logcat output while running the app. If you have set the `logging-severity` option of the Arm NN Delegate to `info`, as shown in this guide, you will see an output similar to the following image.

**Figure 4-1: Arm NN logcat output**



You can set the `logging-severity` level to `debug` to get even more information about which optimization options have been used. If you want no logging at all, do not provide `logging-severity` to the options.

## 5 Related information

Here are some resources related to the material in this guide:

- [Arm Community](#) - ask development questions and find articles and blogs on specific topics from Arm experts.
- [Arm NN Github](#) - raise queries or issues associated with the Arm NN how-to guides.
- [Arm NN Product Documentation](#) - find out more about the latest Arm NN features.

# Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

**Table A-1: First release for version 21.11**

Change	Location
First release	—