

IA1_GRUPO 9 Documentación Técnica

Carne	Nombre
201114502	Rodolfo Orantes Orellana
201800984	Alex Fernando Méndez López

App móvil

- Importar dependencias

```
// Clases para usar la clase CameraX View
implementation "androidx.camera:camera-view:1.3.0-alpha06"
implementation 'com.github.jose-jhr:Library-CameraX:1.0.8'

//clases utilitarias que nos van a permitir en etapa de pre y post
procesamiento
implementation 'org.tensorflow:tensorflow-lite-support:+'
implementation 'org.tensorflow:tensorflow-lite:+'
implementation 'org.tensorflow:tensorflow-lite-metadata:0.1.0'
```

- Configuración del archivo `build.gradle`

```
//no comprima el archivo que contiene el modelo
aaptOptions{
    noCompress = "tflite"
}

buildFeatures{
    viewBinding = true
    mlModelBinding true
}
```

- Clase `MainActivity`

```
class MainActivity : AppCompatActivity() {

    lateinit var binding : ActivityMainBinding
    lateinit var cameraJhr: CameraJhr
    lateinit var classifyImageTf: ClassifyImageTf

    companion object {
```

```

        const val INPUT_SIZE = 224
        const val OUTPUT_SIZE = 3 // Número de etiquetas de salida
    }

    val classes = arrayOf("PERRO", "AVE", "NORMAL")

    //224 x 224
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        //init cameraJHR
        cameraJhr = CameraJhr(this)

        classifyImageTf = ClassifyImageTf(this)
    }

    override fun onWindowFocusChanged(hasFocus: Boolean) {
        super.onWindowFocusChanged(hasFocus)
        if (cameraJhr.allpermissionsGranted() && !cameraJhr.ifStartCamera){
            startCameraJhr()
        }else{
            cameraJhr.noPermissions()
        }
    }

    /**
     * start Camera Jhr
     */
    private fun startCameraJhr() {
        var timeRepeat = System.currentTimeMillis()
        cameraJhr.addListenerBitmap(object : BitmapResponse {
            override fun bitmapReturn(bitmap: Bitmap?) {
                if (bitmap!=null){
                    if (System.currentTimeMillis()>timeRepeat+1000){
                        classifyImage(bitmap)
                        timeRepeat = System.currentTimeMillis()
                    }
                }
            }
        })
        cameraJhr.initBitmap()
        cameraJhr.initImageProxy()

        cameraJhr.start(0,0,binding.cameraPreview,true,false,true)
    }

    private fun classifyImage(img: Bitmap?) {
        val imgReScale = Bitmap.createScaledBitmap(img!!, INPUT_SIZE,
INPUT_SIZE,false)
        classifyImageTf.listenerInterpreter(object :ReturnInterpreter{

```

```

        override fun classify(confidence: FloatArray, maxConfidence: Int)
    {
        binding.txtResult.UiThread("ON ${confidence[0].decimal()} \n
OFF ${confidence[1].decimal()} \n Normal ${confidence[2].decimal()} \n MaxPos
${classes[maxConfidence]}")
    }
    })
    /**Preview Image Get */
    runOnUiThread {
        binding.imgPreview.setImageBitmap(imgReScale)
    }
    classifyImageTf.classify(imgReScale)

}

private fun TextView.UiThread(string: String){
    runOnUiThread {
        this.text = string
    }
}

private fun Float.decimal():String{
    return "%.2f".format(this)
}

}

```

- Layout `activity_main.xml`

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.camera.view.PreviewView
        android:layout_width="match_parent"
        android:id="@+id/camera_preview"
        android:layout_height="match_parent"
        app:scaleType="fillStart"
        >
    </androidx.camera.view.PreviewView>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="resultado"
        android:textAlignment="center"
        android:textSize="22sp"
    >

```

```

        android:id="@+id/txtResult"
        >
    </TextView>

    <ImageView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_alignParentEnd="true"
        android:id="@+id/imgPreview"

        >
    </ImageView>

</RelativeLayout>

```

- **Class** `ClassifyImageTf`

```

import android.content.Context
import android.graphics.Bitmap
import com.ingenieriajhr.teachablemachine.MainActivity.Companion.INPUT_SIZE
import com.ingenieriajhr.teachablemachine.ml.ModelUnquant
import org.tensorflow.lite.DataType
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer
import java.nio.ByteBuffer
import java.nio.ByteOrder

class ClassifyImageTf(context: Context) {

    //get instance model classifier image
    var modelUnquant = ModelUnquant.newInstance(context)

    lateinit var returnInterpreter: ReturnInterpreter

    fun listenerInterpreter(returnInterpreter: ReturnInterpreter){
        this.returnInterpreter = returnInterpreter
    }

    fun classify(img: Bitmap){
        val inputFeature0 = TensorBuffer.createFixedSize(intArrayOf(1,
        INPUT_SIZE, INPUT_SIZE, 3), DataType.FLOAT32)
        val byteBuffer = ByteBuffer.allocateDirect(4* INPUT_SIZE * INPUT_SIZE
        *3)

        byteBuffer.order(ByteOrder.nativeOrder())

        // get 1D array of 224 * 224 pixels in image
        val intValues = IntArray(INPUT_SIZE * INPUT_SIZE)
    
```

```

img!!.getPixels(intValues,0,img!!.width,0,0, img.width,img.height)

// Reemplazar el bucle anidado con operaciones vectorizadas
for (pixelValue in intValues) {
    byteBuffer.putFloat((pixelValue shr 16 and 0xFF) * (1f / 255f))
    byteBuffer.putFloat((pixelValue shr 8 and 0xFF) * (1f / 255f))
    byteBuffer.putFloat((pixelValue and 0xFF) * (1f / 255f))
}
inputFeature0.loadBuffer(byteBuffer)
byteBuffer.clear()

val output = modelUnquant.process(inputFeature0)
val outputFeature = output.outputFeature0AsTensorBuffer
val confidence = outputFeature.floatArray

val maxPos = confidence.indices.maxByOrNull { confidence[it] }?:0

returnInterpreter.classify(confidence, maxPos)
}
}

```

Configuración de Android Studio

SDK Platforms

SDK Platforms SDK Tools SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
<input type="checkbox"/> Android UpsideDownCakePrivacySandbox Preview	UpsideDownCakePrivacySandbox	3	Not installed
<input checked="" type="checkbox"/> Android API 34	34	2	Installed
<input type="checkbox"/> Android API 34	34-ext8	1	Not installed
<input type="checkbox"/> Android API 34	34-ext10	1	Not installed
<input type="checkbox"/> Android TiramisuPrivacySandbox Preview	TiramisuPrivacySandbox	9	Not installed
<input type="checkbox"/> Android 13.0 ("Tiramisu")	33	3	Partially installed
<input type="checkbox"/> Android 13.0 ("Tiramisu")	33-ext4	1	Not installed
<input type="checkbox"/> Android 13.0 ("Tiramisu")	33-ext5	1	Not installed
<input type="checkbox"/> Android 12L ("Sv2")	32	1	Not installed
<input type="checkbox"/> Android 12.0 ("S")	31	1	Not installed
<input type="checkbox"/> Android 11.0 ("R")	30	3	Not installed
<input type="checkbox"/> Android 10.0 ("Q")	29	5	Not installed
<input type="checkbox"/> Android 9.0 ("Pie")	28	6	Not installed
<input type="checkbox"/> Android 8.1 ("Oreo")	27	3	Not installed
<input type="checkbox"/> Android 8.0 ("Oreo")	26	2	Not installed
<input type="checkbox"/> Android 7.1.1 ("Nougat")	25	3	Not installed
<input type="checkbox"/> Android 7.0 ("Nougat")	24	2	Not installed
<input type="checkbox"/> Android 6.0 ("Marshmallow")	23	3	Not installed

☒ Hide Obsolete Packages ☐ Show Package Details

OK Cancel Apply

SDK Tools

SDK Platforms **SDK Tools** SDK Update Sites

Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
<input checked="" type="checkbox"/> Android SDK Build-Tools 34		Installed
<input type="checkbox"/> NDK (Side by side)		Not Installed
<input type="checkbox"/> Android SDK Command-line Tools (latest)		Not Installed
<input type="checkbox"/> CMake		Not Installed
<input type="checkbox"/> Android Auto API Simulators	1	Not installed
<input type="checkbox"/> Android Auto Desktop Head Unit Emulator	2.0	Not installed
<input checked="" type="checkbox"/> Android Emulator	33.1.23	Update Available: 33.1.24
<input checked="" type="checkbox"/> Android Emulator hypervisor driver (installer)	2.0.0	Installed
<input checked="" type="checkbox"/> Android SDK Platform-Tools	34.0.5	Installed
<input type="checkbox"/> Google Play APK Expansion library	1	Not installed
<input type="checkbox"/> Google Play Instant Development SDK	1.9.0	Not installed
<input type="checkbox"/> Google Play Licensing Library	1	Not installed
<input type="checkbox"/> Google Play services	49	Not installed
<input type="checkbox"/> Google USB Driver	13	Not installed
<input type="checkbox"/> Google Web Driver	2	Not installed
<input checked="" type="checkbox"/> Intel x86 Emulator Accelerator (HAXM installer) - Deprecated	7.6.5	Installed
<input type="checkbox"/> Layout Inspector image server for API 29-30	6	Not installed
<input type="checkbox"/> Layout Inspector image server for API 31-34	3	Not installed
<input type="checkbox"/> Layout Inspector image server for API S	3	Not installed

☒ Hide Obsolete Packages ☐ Show Package Details

Machine Learning

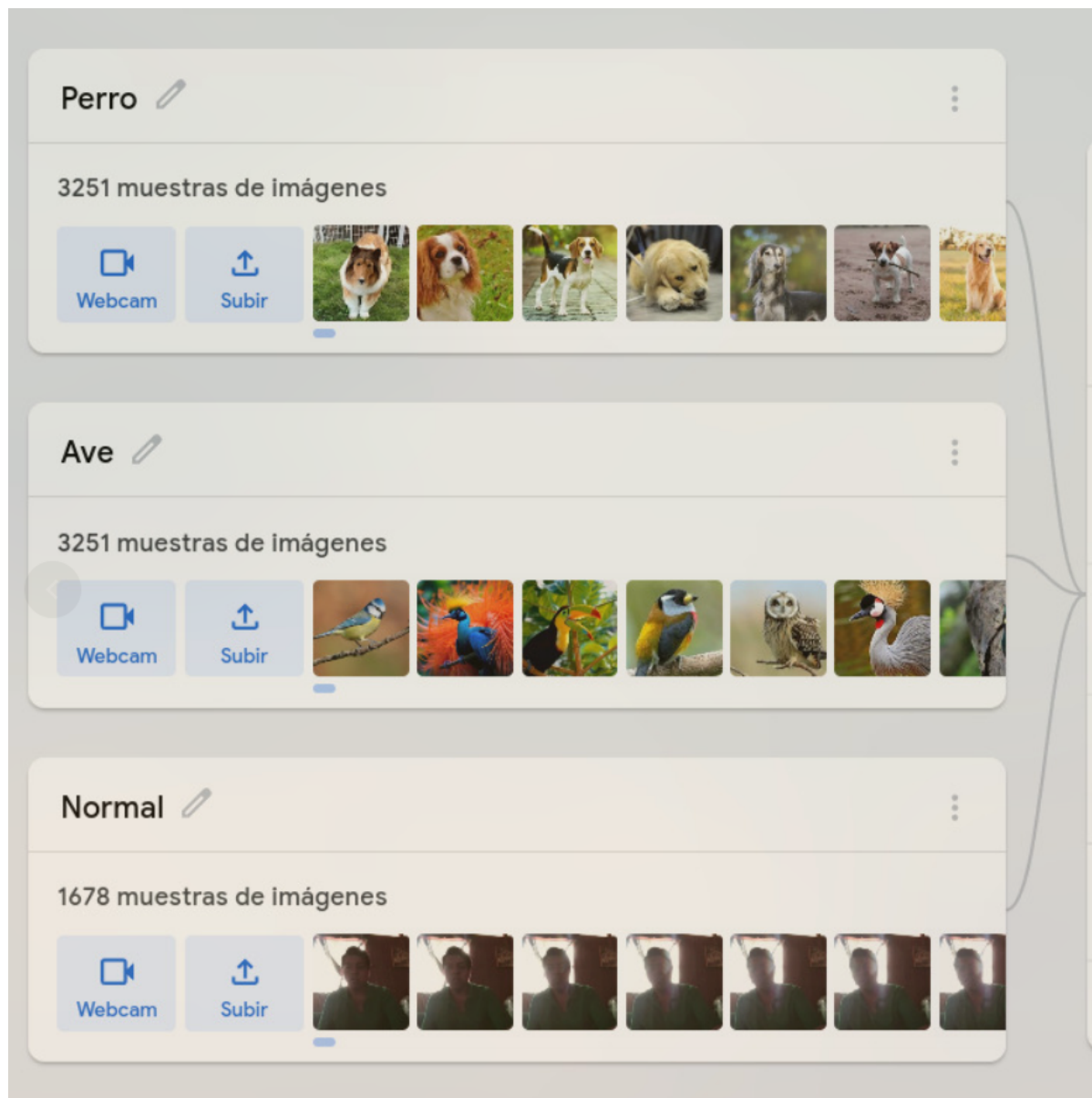
Para crear el modelo de aprendizaje automático se usó la herramienta [Teachable Machine](#).

Insumos

El data set de imágenes que se emplearon para el entramiento se obtuvieron de la siguiente banco de imágenes:

- [Cat & Dog images for Classification](#): Este conjunto de datos abarca una colección de imágenes específicamente seleccionadas con el fin de clasificar entre gatos y perros. Incluye una variedad de imágenes con gatos y perros, diseñadas para ayudar en el entrenamiento de modelos de aprendizaje automático o algoritmos para tareas de clasificación de imágenes. Estas imágenes varían en términos de razas, poses, fondos y otras características relevantes para distinguir entre gatos y perros.
- [BIRDS 525 SPECIES- IMAGE CLASSIFICATION](#): Conjunto de datos de 525 especies de aves. Más de 84635 imágenes de entrenamiento, 2625 imágenes de prueba(5 imágenes por especie) y 2625 imágenes de validación(5 imágenes por especie. Este es un conjunto de datos de muy alta calidad donde solo hay un pájaro en cada imagen y el pájaro generalmente ocupa al menos el 50% de los píxeles de la imagen. Como resultado, incluso un modelo moderadamente complejo logrará precisiones de entrenamiento y prueba en el rango medio del 90. Todas las imágenes son 224 X 224 X 3 imágenes en color en formato jpg.

Carga de insumos en la herramienta de entrenamiento



- Fueron creada 3 clases para entrenar: Perro, Ave y Normal
- En la clase Perro se cargaron 3251 imágenes solamente perros de diferentes razas, fondos y poses con la finalidad de tener buenas precisiones en el entrenamiento y prueba.
- En la clase Ave se cargaron 3251 imágenes solamente aves de diferentes razas, fondos y poses con la finalidad de tener buenas precisiones en el entrenamiento y prueba.
- En la clase Normal se cargaron 1678 imágenes para distinguir un fondo normal o persona distinto a perros y aves.

Configuración y preparación de la herramienta de entrenamiento

- Se configuro 200 épocas.
- El tamaño de lote fueron 16
- La tasa de aprendizaje fue el valor por defecto 0.001

Preparación

Preparando...

02:56 - 65 / 200

Avanzado



Épocas: 200



Tamaño del lote:

16



Tasa de aprendizaje:

0.001



Restablecer valores
predeterminados



Más datos



Pruebas posterior al entrenamiento

Vista previa

Exportar modelo

Entrada

ACTIVADO

Archivo

Selecciona imágenes de tus archivos o arrástralas aquí

Importar imágenes desde Google Drive



Salida

Perro

100%

Ave

Normal

Vista
previa

Exportar modelo

Entrada ☒ ACTIVADO

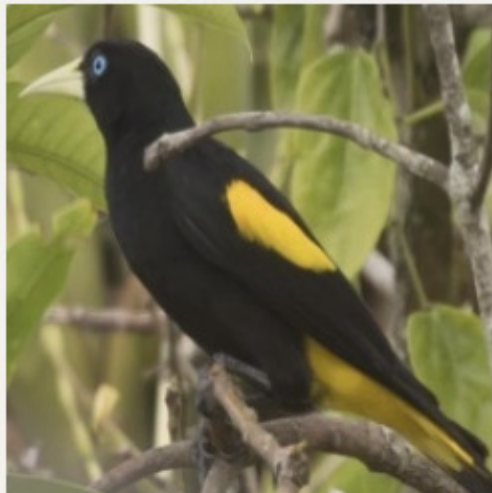
Archivo ▼



Selecciona imágenes de tus
archivos o arrástralas aquí



Importar imágenes desde
Google Drive



Salida

Perro



Ave



100%

Normal



Exportar y descargar el modelo

Exportar el modelo para usarlo en proyectos.

Tensorflow.js

Tensorflow

Tensorflow Lite

Tipo de conversión de modelo:

☒ Punto flotante

☐ Cuantificado

☐ EdgeTPU

Descargar modelo

Convierte tu modelo en un modelo tflite de punto flotante. Ten en cuenta que, aunque la conversión tiene lugar en la nube, solo se sube el modelo preparado, no los datos de preparación.

Fragmentos de código para usar el modelo:

Android

Coral

Contribuye en Github

For this Teachable Machine example, the *Quantized* tflite model is being used. It is using the [TFLite Android example](#), note that the example only supports models with 3 or more classes, even though the classifier itself in the example supports 2.

1. Get the Android app example from [Github](#)

2. Unpack the *converted_tflite_quantized.zip* archive exported from Teachable Machine

3. Copy *converted_tflite_quantized* folder to the example asset folder
`examples/lite/examples/image_classification/android/app/src/main/assets/`

4. Open
`examples/lite/examples/image_classification/android/app/src/main/java/org/tensorflow/lite/examples/classifica`

5. Modify `getModelPath()` and `getLabelPath()` to

@Override

Copiar