LAMPIRAN

LATIH MODEL:

```
!pip install ultralytics==8.0.20
from IPython import display
display.clear_output()
import ultralytics
ultralytics.checks()
from ultralytics import YOLO
from IPython.display import display, Image
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/MyDrive/revisi/data
!ls
%cd /content/drive/MyDrive/revisi/data
!yolo task=detect mode=train model=yolov8n.pt data=
data.yaml epochs=50 imgsz=640 plots=True
```

Data.yml

```
train:
/content/drive/MyDrive/revisi/data/images/train/imag
es
val:
/content/drive/MyDrive/revisi/data/images/valid/imag
es
nc: 5
names:
['Diam','Senyum','Terbelalak','Terpejam','Tertawa']
```

yolo-webcame.py

```
from ultralytics import YOLO
import cv2
import cvzone
import math
cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
model = YOLO("model1.pt")
classNames =
['Terpejam','Terdiam','Terbelalak','Tertawa','Tersenyu
m']
while True:
   success, img = cap.read()
   results = model(img,stream=True)
   for r in results:
      boxes = r.boxes
      for box in boxes:
         # Bounding Box
         x1, y1, x2, y2 = box.xyxy[0]
         x1, y1, x2, y2 = int(x1),int(y1),int(x2),int(y2)
         w, h = x2-x1, y2-y1
         cvzone.cornerRect(img,(x1,y1,w,h))
         # Confidence
         conf = math.ceil((box.conf[0]*100))/100
         # Class Name
         cls = int(box.cls[0])
         cvzone.putTextRect(img,f'{classNames[cls]}
{conf}',(max(0,x1), max(35,y1)))
   cv2.imshow("Image",img)
   cv2.waitKey(1)
```

convert.py

```
from ultralytics import YOLO
# Load the YOLOv8 model
model = YOLO('yolov8nface.pt')
# Export the model to TFLite format
model.export(format='tflite') # creates
'yolov8n_float32.tflite'
# Load the exported TFLite model
tflite_model = YOLO('yolov8n_float32.tflite')
# Run inference
results =
tflite_model('https://ultralytics.com/images/bus.jpg')
```

android:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/
android"
   xmlns:tools="http://schemas.android.com/tools">
   <uses-feature
android:name="android.hardware.camera" />
   <uses-permission
android:name="android.permission.CAMERA" />
   <application
      android:allowBackup="true"
android:dataExtractionRules="@xml/data_extraction_r
ules"
      android:fullBackupContent="@xml/backup_rules"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/Theme.YoloV8TFLite"
      tools:targetApi="31">
      <activity
         android:name=".HomeActivity"
         android:exported="true">
         <intent-filter>
            <action
android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity
         android:name=".MainActivity"
         android:exported="false">
      </activity>
      <uses-native-library
         android:name="libOpenCL.so"
         android:required="false" />
      <uses-native-library
         android:name="libOpenCL-pixel.so"
         android:required="false" />
   </application>
</manifest>
```

```
package com.example.yolov8tflite
data class BoundingBox(
   val x1: Float,
   val y1: Float,
   val x2: Float,
   val y2: Float,
   val cx: Float,
   val cy: Float,
   val w: Float,
   val h: Float,
```

```
    val cnf: Float,
    val cls: Int,
    val clsName: String
)

ackage com.example.yolov8tflite
object Constants {
    const val MODEL_PATH = "best_float32.tflite"
    const val LABELS_PATH = "emosi.txt"
}

package com.example.yolov8tflite
import android.content.Context
import android.graphics.Bitmap
import android.os.SystemClock
import org.tensorflow.lite.DataType
import org.tensorflow.lite.Interpreter
import org.tensorflow.lite.gpu.CompatibilityList
import org.tensorflow.lite.gpu.GpuDelegate
import org.tensorflow.lite.support.common.FileUtil
import
org.tensorflow.lite.support.common.ops.CastOp
import
org.tensorflow.lite.support.common.ops.NormalizeOp
import
org.tensorflow.lite.support.image.ImageProcessor
import org.tensorflow.lite.support.image.TensorImage
import
org.tensorflow.lite.support.tensorbuffer.TensorBuffer
import java.io.BufferedReader
import java.io.IOException
import java.io.InputStream
import java.io.InputStreamReader
class Detector(
    private val context: Context,
    private val modelPath: String,
    private val labelPath: String,
    private val detectorListener: DetectorListener,
) {
    private var interpreter: Interpreter
    private var labels = mutableListOf<String>()
    private var tensorWidth = 0
    private var tensorHeight = 0
    private var numChannel = 0
    private var numElements = 0
    private val imageProcessor =
ImageProcessor.Builder()
        .add(NormalizeOp(INPUT_MEAN,
INPUT_STANDARD_DEVIATION))
        .add(CastOp(INPUT_IMAGE_TYPE))
        .build()
    init {
        val compatList = CompatibilityList()

        val options = Interpreter.Options().apply{
if(compatList.isDelegateSupportedOnThisDevice){
            val delegateOptions =
compatList.bestOptionsForThisDevice
this.addDelegate(GpuDelegate(delegateOptions))
        } else {
            this.setNumThreads(4)
        }
    }
        val model = FileUtil.loadMappedFile(context,
modelPath)
        interpreter = Interpreter(model, options)
        val inputShape =
interpreter.getInputTensor(0)?.shape()
        val outputShape =
interpreter.getOutputTensor(0)?.shape()
        if (inputShape != null) {
            tensorWidth = inputShape[1]
            tensorHeight = inputShape[2]
            // If in case input shape is in format of [1, 3, ...,
...]
            if (inputShape[1] == 3) {
                tensorWidth = inputShape[2]
                tensorHeight = inputShape[3]
            }
        }
        if (outputShape != null) {
            numChannel = outputShape[1]
            numElements = outputShape[2]
        }
        try {
            val inputStream: InputStream =
context.assets.open(labelPath)
            val reader =
BufferedReader(InputStreamReader(inputStream))
            var line: String? = reader.readLine()
            while (line != null && line != "") {
                labels.add(line)
                line = reader.readLine()
            }
            reader.close()
            inputStream.close()
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
    fun restart(isGpu: Boolean) {
        interpreter.close()
        val options = if (isGpu) {
            val compatList = CompatibilityList()
            Interpreter.Options().apply{

if(compatList.isDelegateSupportedOnThisDevice){
                val delegateOptions =
compatList.bestOptionsForThisDevice

this.addDelegate(GpuDelegate(delegateOptions))
            } else {
                this.setNumThreads(4)
            }
        }
        } else {
            Interpreter.Options().apply{
                this.setNumThreads(4)
            }
        }
        val model = FileUtil.loadMappedFile(context,
modelPath)
```

```
        interpreter = Interpreter(model, options)
    }
    fun close() {
        interpreter.close()
    }
    fun detect(frame: Bitmap) {
        if (tensorWidth == 0) return
        if (tensorHeight == 0) return
        if (numChannel == 0) return
        if (numElements == 0) return
        var inferenceTime = SystemClock.uptimeMillis()
        val resizedBitmap =
Bitmap.createScaledBitmap(frame, tensorWidth,
tensorHeight, false)
        val tensorImage =
TensorImage(INPUT_IMAGE_TYPE)
        tensorImage.load(resizedBitmap)
        val processedImage =
imageProcessor.process(tensorImage)
        val imageBuffer = processedImage.buffer
        val output =
TensorBuffer.createFixedSize(intArrayOf(1,
numChannel, numElements), OUTPUT_IMAGE_TYPE)
        interpreter.run(imageBuffer, output.buffer)
        val bestBoxes = bestBox(output.floatArray)
        inferenceTime = SystemClock.uptimeMillis() -
inferenceTime
        if (bestBoxes == null) {
            detectorListener.onEmptyDetect()
            return
        }
        detectorListener.onDetect(bestBoxes,
inferenceTime)
    }
    private fun bestBox(array: FloatArray) :
List<BoundingBox>? {
        val boundingBoxes =
mutableListOf<BoundingBox>()
        for (c in 0 until numElements) {
            var maxConf = CONFIDENCE_THRESHOLD
            var maxIdx = -1
            var j = 4
            var arrayIdx = c + numElements * j
            while (j < numChannel){
                if (array[arrayIdx] > maxConf) {
                    maxConf = array[arrayIdx]
                    maxIdx = j - 4
                }
                j++
                arrayIdx += numElements
            }
            if (maxConf > CONFIDENCE_THRESHOLD) {
                val clsName = labels[maxIdx]
                val cx = array[c] // 0
                val cy = array[c + numElements] // 1
                val w = array[c + numElements * 2]
                val h = array[c + numElements * 3]
                val x1 = cx - (w/2F)
                val y1 = cy - (h/2F)
                val x2 = cx + (w/2F)
                val y2 = cy + (h/2F)
                if (x1 < 0F || x1 > 1F) continue
                if (y1 < 0F || y1 > 1F) continue
                if (x2 < 0F || x2 > 1F) continue
                if (y2 < 0F || y2 > 1F) continue
                boundingBoxes.add(
                    BoundingBox(
                        x1 = x1, y1 = y1, x2 = x2, y2 = y2,
                        cx = cx, cy = cy, w = w, h = h,
                        cnf = maxConf, cls = maxIdx, clsName =
clsName
                    )
                )
            }
        }
        if (boundingBoxes.isEmpty()) return null
        return applyNMS(boundingBoxes)
    }
    private fun applyNMS(boxes: List<BoundingBox>) :
MutableList<BoundingBox> {
        val sortedBoxes = boxes.sortedByDescending {
it.cnf }.toMutableList()
        val selectedBoxes =
mutableListOf<BoundingBox>()

        while(sortedBoxes.isNotEmpty()) {
            val first = sortedBoxes.first()
            selectedBoxes.add(first)
            sortedBoxes.remove(first)

            val iterator = sortedBoxes.iterator()
            while (iterator.hasNext()) {
                val nextBox = iterator.next()
                val iou = calculateIoU(first, nextBox)
                if (iou >= IOU_THRESHOLD) {
                    iterator.remove()
                }
            }
        }
        return selectedBoxes
    }
    private fun calculateIoU(box1: BoundingBox, box2:
BoundingBox): Float {
        val x1 = maxOf(box1.x1, box2.x1)
        val y1 = maxOf(box1.y1, box2.y1)
        val x2 = minOf(box1.x2, box2.x2)
        val y2 = minOf(box1.y2, box2.y2)
        val intersectionArea = maxOf(0F, x2 - x1) *
maxOf(0F, y2 - y1)
        val box1Area = box1.w * box1.h
        val box2Area = box2.w * box2.h
        return intersectionArea / (box1Area + box2Area -
intersectionArea)
    }
    interface DetectorListener {
        fun onEmptyDetect()
        fun onDetect(boundingBoxes: List<BoundingBox>,
inferenceTime: Long)
    }
    companion object {
        private const val INPUT_MEAN = 0f
        private const val INPUT_STANDARD_DEVIATION =
```

```
255f
    private val INPUT_IMAGE_TYPE =
DataType.FLOAT32
    private val OUTPUT_IMAGE_TYPE =
DataType.FLOAT32
    private const val CONFIDENCE_THRESHOLD = 0.5F
    private const val IOU_THRESHOLD = 0.5F
  }
```

```
package com.example.yolov8tflite
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
class HomeActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?)
{
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_home)
    val buttonScan: Button =
findViewById(R.id.button_scan)
    buttonScan.setOnClickListener {
      val intent = Intent(this@HomeActivity,
MainActivity::class.java)
      startActivity(intent)
    }
  }
}
```

```
package com.example.yolov8tflite
import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Bitmap
import android.graphics.Matrix
import android.os.Bundle
import android.util.Log
import android.widget.Button
import
androidx.activity.result.contract.ActivityResultContract
s
import androidx.appcompat.app.AppCompatActivity
import androidx.camera.core.AspectRatio
import androidx.camera.core.Camera
import androidx.camera.core.CameraSelector
import androidx.camera.core.ImageAnalysis
import androidx.camera.core.Preview
import
androidx.camera.lifecycle.ProcessCameraProvider
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import
com.example.yolov8tflite.Constants.LABELS_PATH
import
com.example.yolov8tflite.Constants.MODEL_PATH
import
com.example.yolov8tflite.databinding.ActivityMainBin
ding
import java.util.concurrent.ExecutorService
import java.util.concurrent.Executors
class MainActivity : AppCompatActivity(),
Detector.DetectorListener {
  private lateinit var binding: ActivityMainBinding
  private var isFrontCamera = false
  private var preview: Preview? = null
  private var imageAnalyzer: ImageAnalysis? = null
  private var camera: Camera? = null
  private var cameraProvider:
ProcessCameraProvider? = null
  private var detector: Detector? = null
  private lateinit var cameraExecutor: ExecutorService
  override fun onCreate(savedInstanceState: Bundle?)
{
    super.onCreate(savedInstanceState)
    binding =
ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    // Temukan tombol back_camera menggunakan
ID-nya
    val backButton: Button =
findViewById(R.id.back_camera)
    backButton.setOnClickListener {
      // Buat Intent untuk memulai HomeActivity
      val intent = Intent(this, HomeActivity::class.java)
      startActivity(intent)
    }
    cameraExecutor =
Executors.newSingleThreadExecutor()
    cameraExecutor.execute {
      detector = Detector(baseContext, MODEL_PATH,
LABELS_PATH, this)
    }
    if (allPermissionsGranted()) {
      startCamera()
    } else {
      ActivityCompat.requestPermissions(this,
REQUIRED_PERMISSIONS,
REQUEST_CODE_PERMISSIONS)
    }
    bindListeners()
  }
  private fun bindListeners() {
    binding.apply {
      isGpu.setOnCheckedChangeListener {
buttonView, isChecked ->
        cameraExecutor.submit {
          detector?.restart(isGpu = isChecked)
        }
        if (isChecked) {
buttonView.setBackgroundColor(ContextCompat.getCo
lor(baseContext, R.color.orange))
        } else {
buttonView.setBackgroundColor(ContextCompat.getCo
lor(baseContext, R.color.gray))
        }
      }
      switchCamera.setOnClickListener {
        isFrontCamera = !isFrontCamera
        bindCameraUseCases() // Restart camera with
new lens facing
      }
```

```
    }
  }
  private fun startCamera() {
    val cameraProviderFuture =
ProcessCameraProvider.getInstance(this)
    cameraProviderFuture.addListener({
      cameraProvider = cameraProviderFuture.get()
      bindCameraUseCases()
    }, ContextCompat.getMainExecutor(this))
  }
  private fun bindCameraUseCases() {
    val cameraProvider = cameraProvider ?: throw
IllegalStateException("Camera initialization failed.")
    val rotation = binding.viewFinder.display.rotation
    val cameraSelector = CameraSelector
      .Builder()
      .requireLensFacing(if (isFrontCamera)
CameraSelector.LENS_FACING_FRONT else
CameraSelector.LENS_FACING_BACK)
      .build()
    preview = Preview.Builder()
      .setTargetAspectRatio(AspectRatio.RATIO_4_3)
      .setTargetRotation(rotation)
      .build()
    imageAnalyzer = ImageAnalysis.Builder()
      .setTargetAspectRatio(AspectRatio.RATIO_4_3)
.setBackpressureStrategy(ImageAnalysis.STRATEGY_KE
EP_ONLY_LATEST)
.setTargetRotation(binding.viewFinder.display.rotation
)
.setOutputImageFormat(ImageAnalysis.OUTPUT_IMAG
E_FORMAT_RGBA_8888)
      .build()
    imageAnalyzer?.setAnalyzer(cameraExecutor) {
imageProxy ->
      val bitmapBuffer =
        Bitmap.createBitmap(
          imageProxy.width,
          imageProxy.height,
          Bitmap.Config.ARGB_8888
        )
      imageProxy.use {
bitmapBuffer.copyPixelsFromBuffer(imageProxy.planes
[0].buffer) }
      imageProxy.close()
      val matrix = Matrix().apply
postRotate(imageProxy.imageInfo.rotationDegrees.toFl
oat())
        if (isFrontCamera) {
          postScale(
            -1f,
            1f,
            imageProxy.width.toFloat(),
            imageProxy.height.toFloat()
          )
        }
      }

      val rotatedBitmap = Bitmap.createBitmap(
        bitmapBuffer, 0, 0, bitmapBuffer.width,
bitmapBuffer.height,
        matrix, true
      )
      detector?.detect(rotatedBitmap)
    }
    cameraProvider.unbindAll()
    try {
      camera = cameraProvider.bindToLifecycle(
        this,
        cameraSelector,
        preview,
        imageAnalyzer
      )
preview?.setSurfaceProvider(binding.viewFinder.surfac
eProvider)
    } catch(exc: Exception) {
      Log.e(TAG, "Use case binding failed", exc)
    }
  }
  private fun allPermissionsGranted() =
REQUIRED_PERMISSIONS.all {
    ContextCompat.checkSelfPermission(baseContext,
it) == PackageManager.PERMISSION_GRANTED
  }
  private val requestPermissionLauncher =
registerForActivityResult
ActivityResultContracts.RequestMultiplePermissions())
{
    if (it[Manifest.permission.CAMERA] == true) {
startCamera() }
  }
  override fun onDestroy() {
    super.onDestroy()
    detector?.close()
    cameraExecutor.shutdown()
  }
  override fun onResume() {
    super.onResume()
    if (allPermissionsGranted()){
      startCamera()
    } else {
requestPermissionLauncher.launch(REQUIRED_PERMIS
SIONS)
    }
  }
  companion object {
    private const val TAG = "Camera"
    private const val REQUEST_CODE_PERMISSIONS =
10
    private val REQUIRED_PERMISSIONS =
mutableListOf (
      Manifest.permission.CAMERA
    ).toTypedArray()
  }
  override fun onEmptyDetect() {
    runOnUiThread {
      binding.overlay.clear()
    }
  }
//   override fun onDetect(boundingBoxes:
List<BoundingBox>, inferenceTime: Long) {
//     runOnUiThread {
```

```
//          binding.inferenceTime.text =
"${inferenceTime}ms"
//          binding.overlay.apply {
//              setResults(boundingBoxes)
//              invalidate()
//          }
//      }
//  }
    override fun onDetect(boundingBoxes:
List<BoundingBox>, inferenceTime: Long) {
        runOnUiThread {
            binding.inferenceTime.text =
"${inferenceTime}ms"
            binding.overlay.apply {
                setResults(boundingBoxes)
                invalidate()
            }
        }
    }

}
```

```
package com.example.yolov8tflite
import android.content.Context
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.DashPathEffect
import android.graphics.Paint
import android.graphics.Rect
import android.graphics.Typeface
import android.util.AttributeSet
import android.view.View
import androidx.core.content.ContextCompat

class OverlayView(context: Context?, attrs:
AttributeSet?) : View(context, attrs) {

    private var results = listOf<BoundingBox>()
    private var boxPaint = Paint()
    private var textBackgroundPaint = Paint()
    private var textPaint = Paint()
    private var bounds = Rect()
    init {
        initPaints()
    }
    fun clear() {
        results = listOf()
        textPaint.reset()
        textBackgroundPaint.reset()
        boxPaint.reset()
        invalidate()
        initPaints()
    }
    private fun initPaints() {
        // Text Paint
        textPaint.color = Color.WHITE // Warna teks putih
        textPaint.style = Paint.Style.FILL
        textPaint.textSize = 50f
        textPaint.typeface =
Typeface.create(Typeface.DEFAULT, Typeface.BOLD) //
Font tebal dan stylish
        textPaint.setShadowLayer(4f, 2f, 2f, Color.BLACK)
// Bayangan untuk teks
        // Box Paint
        boxPaint.color =
ContextCompat.getColor(context!!,
R.color.bounding_box_color) // Warna kotak pembatas
hijau
        boxPaint.strokeWidth = 8F
        boxPaint.style = Paint.Style.STROKE
        boxPaint.pathEffect =
DashPathEffect(floatArrayOf(50f, 10f), 0f) // Garis
putus-putus
    }
    override fun draw(canvas: Canvas) {
        super.draw(canvas)

        results.forEach {
            val left = it.x1 * width
            val top = it.y1 * height
            val right = it.x2 * width
            val bottom = it.y2 * height

            // Gambar kotak pembatas terlebih dahulu
            canvas.drawRect(left, top, right, bottom,
boxPaint)
            // Siapkan teks dengan confidence
            val drawableText = "${it.clsName}
(${String.format("%.2f", it.cnf)})"
            // Hitung ukuran teks
            textPaint.getTextBounds(drawableText, 0,
drawableText.length, bounds)
            val textWidth = bounds.width()
            val textHeight = bounds.height()
            // Gambar teks di atas kotak pembatas
            val textX = left
            val textY = top - 10f // Sesuaikan offset vertikal
agar teks berada di atas kotak

            canvas.drawText(drawableText, textX, textY,
textPaint)
        }
    }

    fun setResults(boundingBoxes: List<BoundingBox>) {
        results = boundingBoxes
        invalidate()
    }

    companion object {
        private const val
BOUNDING_RECT_TEXT_PADDING = 8
    }
```

```
package com.example.yolov8tflite
import
androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4
import org.junit.Test
import org.junit.runner.RunWith
import org.junit.Assert.*
/**
```

```
 * Instrumented test, which will execute on an Android
device.
 *
 * See [testing
documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext =
InstrumentationRegistry.getInstrumentation().targetCo
ntext
        assertEquals("com.example.yolov8tflite",
appContext.packageName)
    }
}

package com.example.yolov8tflite

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the
development machine (host).
 *
 * See [testing
documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}

?xml version="1.0" encoding="utf-8"?>
<vector
xmlns:android="http://schemas.android.com/apk/res/
android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path
        android:fillColor="#3DDC84"
        android:pathData="M0,0h108v108h-108z" />
    <path
        android:fillColor="#00000000"
        android:pathData="M9,0L9,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,0L19,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M29,0L29,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M39,0L39,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M49,0L49,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M59,0L59,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M69,0L69,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M79,0L79,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M89,0L89,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M99,0L99,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,9L108,9"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,19L108,19"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,29L108,29"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M0,39L108,39"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
```

```
    android:pathData="M0,49L108,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M0,59L108,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M0,69L108,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M0,79L108,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M0,89L108,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M0,99L108,99"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M19,29L89,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M19,39L89,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M19,49L89,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M19,59L89,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M19,69L89,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M19,79L89,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M29,19L29,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M39,19L39,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M49,19L49,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M59,19L59,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M69,19L69,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
  <path
    android:fillColor="#00000000"
    android:pathData="M79,19L79,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
</vector>

<vector
xmlns:android="http://schemas.android.com/apk/res/
android"
  xmlns:aapt="http://schemas.android.com/aapt"
  android:width="108dp"
  android:height="108dp"
  android:viewportWidth="108"
  android:viewportHeight="108">
  <path android:pathData="M31,63.928c0,0 6.4,-11
12.1,-13.1c7.2,-2.6 26,-1.4 26,-
1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">
    <aapt:attr name="android:fillColor">
      <gradient
        android:endX="85.84757"
        android:endY="92.4963"
        android:startX="42.9492"
        android:startY="49.59793"
        android:type="linear">
        <item
          android:color="#44000000"
          android:offset="0.0" />
        <item
          android:color="#00000000"
          android:offset="1.0" />
      </gradient>
    </aapt:attr>
  </path>
  <path
    android:fillColor="#FFFFFF"
    android:fillType="nonZero"
    android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4
0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -1.1,0.3l-3.9,6.7c-
```

```
6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -
1.1,-0.3C38.8,38.328 38.7,38.828
38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028
31,63.928h46C76.3,56.028 71.8,49.428
65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-
0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-
0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328
43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5
-1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-
0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328
64.6,57.328L64.6,57.328z"
      android:strokeWidth="1"
      android:strokeColor="#00000000" />
</vector>

<vector
xmlns:android="http://schemas.android.com/apk/res/
android"
   android:width="24dp"
   android:height="24dp"
   android:viewportWidth="24.0"
   android:viewportHeight="24.0">
   <path
      android:fillColor="#FFFFFF"
      android:pathData="M18,2h-4.18C12.4,0.84 11.3,0
10,0 8.7,0 7.6,0.84 6.18,2H2c-0.55,0 -1,0.45 -
1,1v16c0,0.55 0.45,1 1,1h16c0.55,0 1,-0.45 1,-1V3c0,-
0.55 -0.45,-1 -1,-1zM10,14.5c-1.93,0 -3.5,-1.57 -3.5,-
3.5s1.57,-3.5 3.5,-3.5 3.5,1.57 3.5,3.5 -1.57,3.5 -
3.5,3.5zM18,8h-2v-2c0,-0.55 -0.45,-1 -1,-1s-1,0.45 -
1,1v2h-2c-0.55,0 -1,0.45 -1,1s0.45,1 1,1h2v2c0,0.55
0.45,1 1,1s1,-0.45 1,-1v-2h2c0.55,0 1,-0.45 1,-1s-0.45,-
1 -1,-1z"/>

</vector>

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


   <TextView
      android:id="@+id/text_app_name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Deteksi Wajah"
      android:textSize="24sp"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="50dp"/>

   <Button
      android:id="@+id/button_scan"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Kamera"

      android:layout_below="@id/text_app_name"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="20dp"/>

</RelativeLayout>

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
android:background="@color/black"

xmlns:android="http://schemas.android.com/apk/res/
android"
   xmlns:app="http://schemas.android.com/apk/res-
auto"
   android:id="@+id/camera_container"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   xmlns:tools="http://schemas.android.com/tools">

   <androidx.camera.view.PreviewView
      android:id="@+id/view_finder"
      android:layout_width="0dp"
      android:layout_height="0dp"

app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintDimensionRatio="3:4"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.5"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.5"
      app:scaleType="fillStart" />

   <com.example.yolov8tflite.OverlayView
      android:id="@+id/overlay"
      android:layout_width="0dp"
      android:layout_height="0dp"
      android:translationZ="5dp"

app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintDimensionRatio="3:4"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.5"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.5" />

   <TextView
      tools:text="100ms"
      android:textColor="@color/white"
      android:id="@+id/inferenceTime"
      android:layout_margin="32dp"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

   <androidx.constraintlayout.widget.ConstraintLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="@color/white"
      android:paddingVertical="4dp"
```

```
app:layout_constraintBottom_toBottomOf="parent">

    <Button
      android:id="@+id/back_camera"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="home"
      app:layout_constraintVertical_bias="0.5"/>

    <ToggleButton
      android:id="@+id/isGpu"
      android:background="@color/orange"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textColor="@color/white"
      android:textOff="@string/gpu"
      android:textOn="@string/gpu"

app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.5" />

    <Button
      android:id="@+id/switch_camera"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="switch"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintTop_toTopOf="parent"

app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintVertical_bias="0.5"/>

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon
xmlns:android="http://schemas.android.com/apk/res/
android">
  <background
android:drawable="@drawable/ic_launcher_backgrou
nd" />
  <foreground
android:drawable="@drawable/ic_launcher_foregroun
d" />
  <monochrome
android:drawable="@drawable/ic_launcher_foregroun
d" />
</adaptive-icon>

?xml version="1.0" encoding="utf-8"?>
<adaptive-icon
xmlns:android="http://schemas.android.com/apk/res/
android">
  <background
android:drawable="@drawable/ic_launcher_backgrou
nd" />
  <foreground
android:drawable="@drawable/ic_launcher_foregroun
d" />
  <monochrome
android:drawable="@drawable/ic_launcher_foregroun
d" />
</adaptive-icon>

?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="black">#FF000000</color>
  <color name="white">#FFFFFFFF</color>
  <color
name="bounding_box_color">#66BB6A</color>
  <color name="red">#ff0000</color>
  <color name="orange">#F93</color>
  <color
name="black_transparent">#80000000</color>
</resources>

<resources>
  <string name="app_name">Deteksi Wajah</string>
  <string name="gpu">GPU</string>
</resources>

<resources
xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <style name="Base.Theme.YoloV8TFLite"
parent="Theme.Material3.DayNight.NoActionBar">
    <!-- Customize your light theme here. -->
    <!-- <item
name="colorPrimary">@color/my_light_primary</ite
m> -->
  </style>

  <style name="Theme.YoloV8TFLite"
parent="Base.Theme.YoloV8TFLite" />
</resources>

<resources
xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <style name="Base.Theme.YoloV8TFLite"
parent="Theme.Material3.DayNight.NoActionBar">
    <!-- Customize your dark theme here. -->
    <!-- <item
name="colorPrimary">@color/my_dark_primary</ite
m> -->
  </style>
</resources>

val model = Best1Float32.newInstance(context)

// Creates inputs for reference.

val inputFeature0 =
TensorBuffer.createFixedSize(intArrayOf(1, 640, 640,
3), DataType.FLOAT32)

inputFeature0.loadBuffer(byteBuffer)
```

```
// Runs model inference and gets result.

val outputs = model.process(inputFeature0)

val outputFeature0 =
outputs.outputFeature0AsTensorBuffer

// Releases model resources if no longer used.

model.close()try {

   Best1Float32 model =
Best1Float32.newInstance(context);

   // Creates inputs for reference.

   TensorBuffer inputFeature0 =
TensorBuffer.createFixedSize(new int[]{1, 640, 640, 3},
DataType.FLOAT32);

   inputFeature0.loadBuffer(byteBuffer);

   // Runs model inference and gets result.

   Best1Float32.Outputs outputs =
model.process(inputFeature0);

   TensorBuffer outputFeature0 =
outputs.getOutputFeature0AsTensorBuffer();

   // Releases model resources if no longer used.

   model.close();

} catch (IOException e) {

   // TODO Handle the exception

}
```

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

PROPOSAL

Mahasiswa : NUR ANJANI	Pembimbing I : Muhammad Basri, S.T.,M.T.
NIM : 220280020	Pembimbing II : WAHYUDDIN,S.Kom.,M.Kom
Judul Skripsi : APLIKASI DETEKSI EMOSI PADA WAJAH DENGAN MESIN LEARNING	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1		Konsultasi 1 - Ikuti panduan - Teori pada BAB di perjelas	
Konsultasi 2		Konsultasi 2 - Teori Algoritma harus di masukkan di BAB II	
Konsultasi 3		Konsultasi 3 - Desain sistem diperbaiki - Kata asing dimiringkan	
Konsultasi 4 Acc proposal.		Konsultasi 4 Acc	
Konsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah...

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib ditampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 28 Maret 2024

Mengetahui
Ketua Program Studi

Marlina, S.Kom.,M.Kom.
NBM. 1162 680

Mahasiswa

Nur Anjani
NIM. 220280020

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : Nur Anjani	Pembimbing I : Muh. Basri, ST., MT
NIM : 220280020	Pembimbing II : Wahyuddin, S.Kom., M.Kom
Judul Skripsi : APLIKASI DETEKSI EKSPRESI PADA WAJAH DENGAN MESIN LEARNING	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1		Konsultasi 1	
Konsultasi 2		Konsultasi 2	
Konsultasi 3		Konsultasi 3	
Konsultasi 4		Konsultasi 4	
Konsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah...

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan ...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 2024

Mengetahui
Ketua Program Studi

Marlina, S.Kom.,M.Kom.
NBM. 1162 680

Mahasiswa

Nur Anjani
NIM. 220280020

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan akhir dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas kartun berwarna Ujian muda dan dicetak timbal balik

SERTIFIKAT

Nomor : 308/PRK/LB-TI/II.3.AU/D/2023

Diberikan Kepada :

Nur Anjani

NIM.220280020

Telah Menyelesaikan

PRAKTIKUM III

Laboratorium Teknik Informatika
Program Studi Teknik Informatika Fakultas Teknik
Universitas Muhammadiyah Parepare

Pada tanggal 16 Januari 2023 - 29 Maret 2023

Parepare, 15 Juni 2023

KETUA PRODI
TEKNIK INOFORMATIKA

Marlina. S.kom., M.kom
NBM. 1162 680

KEPALA
LABORATORIUM

Ir. Untung Suwardoyo, S.Kom., MT., IPP.
NBM. 1288 973

SERTIFIKAT

Nomor :136/PRK/LB-TI/II.3.AU/D/2022

diberikan kepada :

NUR ANJANI

NIM.220280029

telah menyelesaikan

PRAKTIKUM IV

Laboratorium Teknik Informatika
Program Studi Teknik Informatika Fakultas Teknik
Universitas Muhammadiyah Parepare

Pada tanggal 14 September 2022 - 25 Desember 2022

Parepare, 02 Januari 2023

KETUA PRODI
TEKNIK INFORMATIKA

MARLINA, S.Kom., M.Kom

KEPALA
LABORATORIUM

Ir. UNTUNG SUWARDOYO, S.Kom., MT.,IPP.

SERTIFIKAT

Nomor : 735/PRK/LB-TI/II.3.AU/D/2023

Diberikan Kepada :

Nur Anjani

NIM.220280020

Telah Menyelesaikan

PRAKTIKUM V

Laboratorium Teknik Informatika
Program Studi Teknik Informatika Fakultas Teknik
Universitas Muhammadiyah Parepare

Pada tanggal 18 September - 16 Desember 2023

Parepare, 03 Januari 2024

KETUA PRODI
TEKNIK INOFORMATIKA

Marlina. S.kom., M.kom
NBM. 1162 680

KEPALA
LABORATORIUM

Ir. Untung Suwardoyo, S.Kom., MT., IPP.
NBM. 1288 973