# Rapport SAE 501

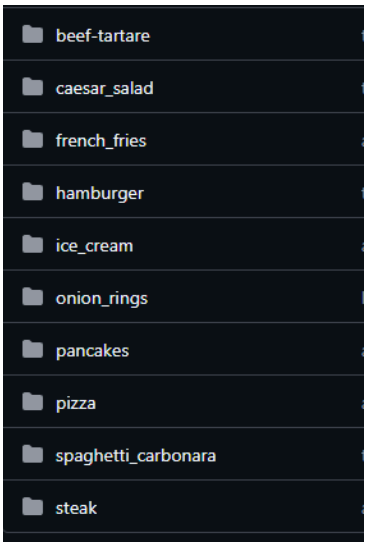
## Semaine 10

MALORON Arthur,
DOTTO Matis,
MANICK Luc,
BELMADANI Abdourrahmane

# Sujets abordées

## Entrainement du modèle

Nous avons continué à tenter de faire fonctionner une installation docker, mais sans succès jusqu'à maintenant.




Nous avons également créé 10 classes d'images de plats, avec 100 images chacune, le tout labellisées.

## Stockage des images dans l'API

Nous avons ajouté une fonctionnalité permettant de stocker les images dans notre API, envoyées depuis l'application mobile. Pour tester si les images se stockent correctement, il nous manque seulement les identifiants pour accéder à la machine virtuelle afin d'y héberger l'API.

## Ajout du bouton qui permet de prendre et d'enregistrer une photo.

### 1. Création de l'Overlay (surimpression)

La fonction `createOverlayBitmap` génère un bitmap transparent (overlay) de la même taille que la photo capturée. Ce bitmap sera utilisé pour dessiner les objets détectés (par exemple, des objets détectés par un modèle d'intelligence artificielle) avec des boîtes autour d'eux et des informations textuelles sur le type d'objet et la certitude de la détection.

- **Facteurs d'échelle** : Les coordonnées des objets détectés sont ajustées en fonction de l'échelle de l'image capturée par rapport à la taille de la prévisualisation de la caméra.
- **Rectangle et texte** : Un rectangle vert est dessiné autour de chaque objet détecté, et un texte rouge (avec le nom de l'objet et la certitude) est ajouté juste au-dessus du rectangle.

```
private fun createOverlayBitmap(photoWidth: Int, photoHeight: Int): Bitmap {
    val overlayBitmap = Bitmap.createBitmap(photoWidth, photoHeight, Bitmap.Config.ARGB_8888)
    val canvas = android.graphics.Canvas(overlayBitmap)

    val scaleX = photoWidth.toFloat() / cameraPreviewSize.value.width
    val scaleY = photoHeight.toFloat() / cameraPreviewSize.value.height

    for (detectedObject in detectedObjects) {
        val rectPaint = android.graphics.Paint().apply {
            color = android.graphics.Color.GREEN
            strokeWidth = 5f
            style = android.graphics.Paint.Style.STROKE
        }
        val textPaint = android.graphics.Paint().apply {
            color = android.graphics.Color.RED
            textSize = 40f
        }

        // Ajustez les coordonnées des rectangles en fonction des facteurs d'échelle
        val scaledLeft = detectedObject.box.left * scaleX
        val scaledTop = detectedObject.box.top * scaleY
        val scaledRight = detectedObject.box.right * scaleX
        val scaledBottom = detectedObject.box.bottom * scaleY

        // Dessinez le rectangle ajusté
        canvas.drawRect(scaledLeft, scaledTop, scaledRight, scaledBottom, rectPaint)

        // Dessinez le texte au-dessus du rectangle
        canvas.drawText(
            "${detectedObject.name}: ${"%.2f".format(detectedObject.certainty * 100)}%",
            scaledLeft,
            scaledTop - 10,
            textPaint
        )
    }

    return overlayBitmap
}
```

## 2. Combinaison de l'Image Capturée et de l'Overlay

La fonction `combineBitmaps` permet de superposer l'overlay sur l'image capturée. Elle crée un nouveau bitmap de la même taille que l'image capturée et y dessine à la fois l'image originale et l'overlay.

```
private fun combineBitmaps(capturedBitmap: Bitmap, overlayBitmap: Bitmap): Bitmap {
    // Créer un nouveau Bitmap avec la même taille que l'image capturée
    val combinedBitmap = Bitmap.createBitmap(capturedBitmap.width, capturedBitmap.height, capturedBitmap.config)

    // Dessiner l'image capturée et l'overlay
    val canvas = android.graphics.Canvas(combinedBitmap)
    canvas.drawBitmap(capturedBitmap, 0f, 0f, null)
    canvas.drawBitmap(overlayBitmap, 0f, 0f, null)

    return combinedBitmap
}
```

## 3. Capture et Sauvegarde de la Photo avec Overlay

La fonction `capturePhotoWithOverlay` prend la photo avec l'overlay et l'enregistre dans la galerie de l'appareil. Un nom de fichier unique est généré pour chaque photo, et elle est enregistrée dans le répertoire "/storage/emulated/0/Pictures/S501" sur le stockage externe.

- La photo capturée est ensuite combinée avec l'overlay, et l'image résultante est enregistrée.
- Si la capture échoue, un message d'erreur est affiché à l'utilisateur.

```
private fun capturePhotoWithOverlay(controller: LifecycleCameraController) {
    val fileName = "IMG_${SimpleDateFormat("yyyyMMdd_HHmmss", Locale.US).format(System.currentTimeMillis())}.jpg"
    val contentValues = ContentValues().apply {
        put(MediaStore.Images.Media.DISPLAY_NAME, fileName)
        put(MediaStore.Images.Media.MIME_TYPE, "image/jpeg")
        put(MediaStore.Images.Media.RELATIVE_PATH, "Pictures/S501")
    }
    val uri = contentResolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, contentValues)

    uri?.let { uri ->
        controller.takePicture(
            ImageCapture.OutputFileOptions.Builder(contentResolver.openOutputStream(uri)!!).build(),
            ContextCompat.getMainExecutor(applicationContext),
            object : ImageCapture.OnImageSavedCallback {
                override fun onImageSaved(output: ImageCapture.OutputFileResults) {
                    val capturedImageUri = output.savedUri ?: uri
                    val capturedBitmap = BitmapFactory.decodeStream(contentResolver.openInputStream(capturedImageUri))

                    // Validate capturedBitmap
                    if (capturedBitmap == null) {
                        Log.e("Capture", "Failed to load captured bitmap")
                        Toast.makeText(applicationContext, "Erreur lors de la capture de l'image", Toast.LENGTH_SHORT).show()
                        return
                    }

                    // Create the overlay bitmap with correct dimensions
                    val overlayBitmap = createOverlayBitmap(capturedBitmap.width, capturedBitmap.height)

                    // Combine the captured photo with the overlay
                    val combinedBitmap = combineBitmaps(capturedBitmap, overlayBitmap)

                    saveCombinedImage(combinedBitmap, capturedImageUri)

                    Toast.makeText(applicationContext, "Photo enregistrée avec overlay!", Toast.LENGTH_SHORT).show()
                }

                override fun onError(exception: ImageCaptureException) {
                    Toast.makeText(applicationContext, "Erreur lors de l'enregistrement!", Toast.LENGTH_SHORT).show()
                }
            }
        )
    }
}
private fun saveCombinedImage(combinedBitmap: Bitmap, uri: Uri) {
```

## 4. Sauvegarde de l'Image Combinée

Une fois que l'image combinée est créée (photo + overlay), elle est sauvegardée dans le stockage externe avec la fonction `saveCombinedImage`.

```
private fun saveCombinedImage(combinedBitmap: Bitmap, uri: Uri) {
    val outputStream = contentResolver.openOutputStream(uri)
    if (outputStream != null) {
        combinedBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)
        outputStream.close()
        Log.d("Save", "Image saved to $uri")
    } else {
        // Handle the error when the outputStream is null
        Toast.makeText(applicationContext, "Erreur lors de l'enregistrement de l'image!", Toast.LENGTH_SHORT).show()
        Log.e("Save", "Failed to obtain output stream for URI: $uri")
    }
}
```

## 5. Interface Utilisateur : Bouton de Capture

Un bouton flottant en bas de l'écran permet à l'utilisateur de capturer la photo avec l'overlay. Ce bouton utilise une icône de caméra ("📸") et déclenche la fonction `capturePhotoWithOverlay` lorsqu'il est cliqué.

```
Box(
    modifier = Modifier
        .align(Alignment.BottomCenter)
        .padding(16.dp)
) {
    Box(
        modifier = Modifier
            .size(80.dp)
            .background(Color.Transparent, shape = CircleShape)
            .border(5.dp, Color.White, CircleShape)
            .align(Alignment.Center)
    ) {
        Button(
            onClick = {
                capturePhotoWithOverlay(controller)
            },
            modifier = Modifier
                .fillMaxSize(),
            shape = CircleShape,
            colors = ButtonDefaults.buttonColors(
                containerColor = Color.Transparent
            )
        ) {
            Text(
                text = "📸",
                color = Color.White,
                fontSize = 24.sp
            )
        }
    }
}
```

# Objectifs hebdomadaires

- Réussir à avoir un environnement pouvant utiliser les scripts du github de tensorflow ou trouver une alternative

## Lien du Trello

https://trello.com/invite/b/670e75ea832d0d8d7b7625cf/ATTI30db0fce264f5fcced5a98e25047eb67DF2B89AE/s501,