Studenti: Stefano Arasi, Marko P’yeshchyk

# Android

## Introduzione

Questa applicazione sviluppata in Kotlin per android ha come scopo l’utilizzo della telecamera per la ripresa di un video di un utente e l’applicazione di azioni su di un Video Player per permettere la visione senza bisogno dell’utilizzo di un input diretto da parte dell’utente.

Una volta che il video registrato è stato ripreso viene analizzato da una rete neurale per riconoscere quale dei movimenti tra quelli utilizzati per il suo addestramento è stato eseguito e verrà eseguita una azione a seconda del risultato.

Le azioni disponibili con la corrispondente azione sono:

* Fare il pollice in su: aumenta il volume del dispositivo.
* Fare il pollice in giù: diminuisce il volume del dispositivo.
* Fare un simbolo di stop con il palmo della mano: mettere in pausa il video (o riprendere la riproduzione se già in pausa)
* Muovere la mano verso destra: manda il video avanti di 10 secondi.
* Muovere la mano verso sinistra: manda il video indietro di 10 secondi.

## Implementazione

### Pattern di design

L’applicazione è stata sviluppata secondo il pattern di design MVC (Model View Controller) perché è un'architettura software molto usata, specialmente quelle con interfaccia utente. Il suo scopo principale è separare la logica di presentazione dalla logica di business, migliorando la manutenibilità, testabilità e riusabilità del codice.

### Controller

Il controller rappresenta il punto di connessione tra dati ed interfaccia grafica dell’applicazione, nel caso della nostra applicazione esegue anche l’inizializzazione del sistema.

L’inizializzazione dell’applicazione viene eseguito nel seguente modo:

class MainActivity : AppCompatActivity() {  
 private lateinit var volumeManager:VolumeManager;  
 private lateinit var cameraManager: CameraManager;  
 private lateinit var videoManager: VideoManager;  
  
 private val REQUIRED\_PERMISSIONS = *arrayOf*(  
 Manifest.permission.*CAMERA*,  
 Manifest.permission.*READ\_MEDIA\_VIDEO*  
)  
 private val REQUEST\_CODE\_PERMISSIONS = 10  
  
 //set up  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_main*)  
  
 *requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*  
  
volumeManager = VolumeManager(*applicationContext*)  
  
 cameraManager = CameraManager(this, findViewById(R.id.*previewView*))  
  
 videoManager = VideoManager(this, findViewById(R.id.*videoView*));  
  
  
 if (allPermissionsGranted()) {  
 cameraManager.startCamera()  
 } else {  
 ActivityCompat.requestPermissions(this, REQUIRED\_PERMISSIONS, REQUEST\_CODE\_PERMISSIONS)  
 }  
  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->**  
val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}**  
}  
  
 private fun allPermissionsGranted() = REQUIRED\_PERMISSIONS.*all* **{**  
ContextCompat.checkSelfPermission(this, **it**) == PackageManager.*PERMISSION\_GRANTED*  
**}**  
  
  
override fun onRequestPermissionsResult(  
 requestCode: Int,  
 permissions: Array<out String>,  
 grantResults: IntArray  
 ) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
  
 if (requestCode == REQUEST\_CODE\_PERMISSIONS) {  
 if (grantResults.*isNotEmpty*() && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*  
&& grantResults[1] == PackageManager.*PERMISSION\_GRANTED*) {  
 // Both permissions granted: Start Camera & Play Video  
 //startCamera()  
 } else {  
 // If any permission is denied, show a toast message and close the app  
 Toast.makeText(this, "All permissions are required", Toast.*LENGTH\_SHORT*).show()  
 finish()  
 }  
 }  
 }

}

Inizialmente vengono definiti gli oggetti che faranno riferimento alle classi del Model per la gestione della camera, del video player e del volume.

Poi vengono definiti i permessi da richiedere, CAMERA per accedere alla camera, READ\_MEDIA\_VIDEO.

La funzione onCreate() viene richiamata come prima cosa nell’applicazione.

Questa prepara il layout dell’applicazione e fissa il layout dell’applicazione a verticale.

Poi vengono inizializzati i tre manager descritti precedentemente.

Viene poi controllato se i permessi sono già stati concessi, se sono concessi viene attivata la camera, altrimenti richiede i permessi all’utente.

Nel caso i permessi vengano negati l’applicazione viene chiusa.

Come ultima cosa fa in modo che l’applicazione non venga coperta dalla barra di stato, barra di navigazione, etc.

Vengono poi definite le seguenti funzioni per gestire l’interazione dell’utente con l’ui:

private val PICK\_VIDEO\_REQUEST = 1  
  
fun onPickVideo(view: View) {  
 val intent = Intent(Intent.*ACTION\_PICK*)  
 intent.*type* = "video/\*"  
 startActivityForResult(intent, PICK\_VIDEO\_REQUEST)  
}  
  
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
 super.onActivityResult(requestCode, resultCode, data)  
  
 if (requestCode == PICK\_VIDEO\_REQUEST && resultCode == *RESULT\_OK* && data != null) {  
 val videoUri: Uri? = data.*data*  
videoUri?.*let* **{** videoManager.loadAndPlayVideo(**it**) **}**  
}  
}

Viene inizialmente richiesto all’utente di scegliere un video, una volta scelto viene ricavato il suo uri e passato al [CameraManager](#_CameraManager) per riprodurre quel video.

fun startOrStopRecording(view: View) {  
  
 cameraManager.startOrStopRecording **{** uri **->**  
if (uri == null) {  
 Toast.makeText(this, "Something went wrong while tryng to record the video", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**  
}

Questa funzione fa partire o ferma la registrazione avviata attraverso [CameraManager](#_CameraManager), nel caso ci sia un errore viene fatto visualizzare all’utente.

DA COLLEGARE

//Neural Network applicable functions  
  
fun pauseVideo(view: View) {  
 videoManager.togglePlayPause()  
}  
  
fun aheadVideo(view: View) {  
 videoManager.seekForward()  
}  
  
fun behindVideo(view: View) {  
 videoManager.seekBackward()  
}  
  
  
fun lowerVolume(){  
 volumeManager.lowerVolume();  
}  
  
fun raiseVolume(){  
 volumeManager.raiseVolume();  
}

### Model

Il modello rappresenta i dati e la logica di business dell’applicazione.

Nel nostro caso comprende 3 classi: CameraManager, VideoManager, VolumeManager.

#### CameraManager

È la classe che gestisce l’utilizzo della camera nell’applicazione.

La libreria utilizzata per la camera è CameraX, ufficiale di android.

Per poterla utilizzare nell’applicazione è necessario implementare la libreria nel file gradle in questo modo:

// CameraX core library using the camera2 implementation  
val camerax\_version = "1.5.0-alpha06"  
// The following line is optional, as the core library is included indirectly by camera-camera2  
*implementation*("androidx.camera:camera-core:${camerax\_version}")  
*implementation*("androidx.camera:camera-camera2:${camerax\_version}")  
// If you want to additionally use the CameraX Lifecycle library  
*implementation*("androidx.camera:camera-lifecycle:${camerax\_version}")  
// If you want to additionally use the CameraX VideoCapture library  
*implementation*("androidx.camera:camera-video:${camerax\_version}")  
// If you want to additionally use the CameraX View class  
*implementation*("androidx.camera:camera-view:${camerax\_version}")  
// If you want to additionally add CameraX ML Kit Vision Integration  
*implementation*("androidx.camera:camera-mlkit-vision:${camerax\_version}")  
// If you want to additionally use the CameraX Extensions library  
*implementation*("androidx.camera:camera-extensions:${camerax\_version}")

Inoltre, per poterlo utilizzare è necessario ricevere i permessi di utilizzare la telecamera da parte dell’utente per motivi di sicurezza.

Una volta che l’utente permette l’utilizzo della camera una istanza di CameraX viene creata attraverso la seguente funzione:

class CameraManager(private val context: Context, private val previewView: PreviewView) {  
 private var videoCapture:VideoCapture<Recorder>?=null;  
 private var currentRecording:Recording?=null;  
  
 fun startCamera() {  
 val cameraProviderFuture = ProcessCameraProvider.getInstance(context)  
  
 cameraProviderFuture.addListener(**{**  
val cameraProvider = cameraProviderFuture.get()  
  
 val preview = Preview.Builder().build().*also***{**  
 **it**.setSurfaceProvider(previewView.*surfaceProvider*)  
 **}**  
  
val recorder = Recorder.Builder()  
 .setQualitySelector(QualitySelector.from(Quality.*HIGHEST*))  
 .build()  
  
 videoCapture = VideoCapture.withOutput(recorder)  
  
 val cameraSelector = CameraSelector.*DEFAULT\_BACK\_CAMERA*  
  
try {  
 cameraProvider.unbindAll()  
 cameraProvider.bindToLifecycle(  
 context as androidx.lifecycle.LifecycleOwner, cameraSelector, preview, videoCapture  
 )  
 } catch (exc:Exception){  
 Log.e("VideoManager", "Use case binding failed", exc)  
 }  
 **}**,ContextCompat.getMainExecutor(context))  
 }

}

Innanzi tutto, viene creata un’istanza di ProcessCameraProvider che gestisce i casi d’uso di CameraX e restituirà una CameraProvider una volta pronto.

Poi viene creato l’oggetto preview che crea il caso d’uso preview video e lo collega alla PreviewView, in modo che la fotocamera venga mostrata a schermo.

Vengono poi creati un recorder per poter registrare un video e un videoCapture che lo utilizza.

Viene poi selezionata la camera posteriore.

Infine, disconnette qualsiasi precedente caso d’uso e collega la fotocamera al ciclo di vita di context, avviando la preview e preparando la registrazione.

La prossima funzione viene richiamata quando il pulsante per registrare viene premuto, per poter salvare il video che verrà registrato l’utente dovrà garantire i permessi di accesso alla memoria del dispositivo:

fun startOrStopRecording(onVideoSaved:(Uri?) ->Unit)  
  
{  
 val videoCapture = videoCapture ?:return  
  
 if (currentRecording != null) {  
 currentRecording ?.stop()  
 currentRecording = null  
 Toast.makeText(context, "Recording stopped", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 val name = "video\_${System.currentTimeMillis()}.mp4"  
  
 val contentValues = ContentValues().*apply* **{**  
put(MediaStore.MediaColumns.*DISPLAY\_NAME*, name)  
 put(MediaStore.MediaColumns.*MIME\_TYPE*, "video/mp4")  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 put(MediaStore.MediaColumns.*RELATIVE\_PATH*, "Movies/MyAppVideos")  
 }  
 **}**  
  
val mediaStoreOutputOptions = MediaStoreOutputOptions.Builder(  
 context.*contentResolver*,  
 MediaStore.Video.Media.*EXTERNAL\_CONTENT\_URI*  
)  
 .setContentValues(contentValues)  
 .build()  
  
 currentRecording = videoCapture.*output*  
.prepareRecording(context as androidx.activity.ComponentActivity, mediaStoreOutputOptions)  
 .start(ContextCompat.getMainExecutor(context)) **{**  
recordEvent **->**  
when(recordEvent) {  
 is VideoRecordEvent.Start -> {  
 Toast.makeText(context, "Recording started", Toast.*LENGTH\_SHORT*).show()  
 }  
 is VideoRecordEvent.Finalize -> {  
 if (!recordEvent.hasError()) {  
 val uri = recordEvent.*outputResults*.*outputUri*  
Toast.makeText(context, "Video saved to $uri", Toast.*LENGTH\_LONG*).show()  
 onVideoSaved(uri)  
 } else {  
 Toast.makeText(context, "Error saving video: ${recordEvent.*error*}", Toast.*LENGTH\_SHORT*).show()  
 }  
 currentRecording = null  
 }  
 }  
 **}**  
}  
}

Questa funzione serve a gestire l’avvio o l’arresto della registrazione video usando la libreria CameraX, se la registrazione è in corso, la ferma; se non è in corso, la avvia.

Come prima cosa controlla se ci sia già una registrazione in corso, in quel caso la ferma.

Altrimenti, viene creato un nome unico per il video registrato e prepara i metadati per salvare il file.

Poi definisce un oggetto MediaStoreOutputOptions per definire dove e come salvare il video.

Viene poi avviata la registrazione, in particolare:

* Usa videoCapture.output.prepareRecording(...) per preparare la registrazione.
* Chiama start(...) per iniziarla.
* Registra recordEvent per reagire agli eventi della registrazione.

Nel caso la registrazione venga iniziata notificato all’utente di ciò.

Nel caso la registrazione venga terminata:

* Se è andato tutto correttamente, viene mostrato all’utente dove il file viene salvato e viene richiamato onVideoSaved() per salvarlo.
* Se c’è stato un errore l’utente viene notificato di ciò.

#### VideoManager

È la classe per la gestione del Video Player dato l’uri del video da riprodurre.

class VideoManager (private val context: Context, private val videoView: VideoView) {  
  
 fun loadAndPlayVideo(uri: Uri) {  
 val mediaController = MediaController(context)  
 mediaController.setAnchorView(videoView)  
 videoView.setMediaController(mediaController)  
  
 videoView.setVideoURI(uri)  
 videoView.setOnPreparedListener **{** mp: MediaPlayer **->**  
  
val videoWidth = mp.*videoWidth*  
val videoHeight = mp.*videoHeight*  
  
if (videoWidth == 0 || videoHeight == 0) {  
 videoView.start()  
 return@setOnPreparedListener  
 }  
  
 val parent = videoView.*parent* as View  
 val parentWidth = parent.*width*  
  
val aspectRatio = videoHeight.toFloat() / videoWidth.toFloat()  
 var calculatedHeight = (parentWidth \* aspectRatio).toInt()  
  
 // Calculate 40% of screen height in pixels  
 val displayMetrics = context.*resources*.*displayMetrics*  
val maxHeightPx = (displayMetrics.heightPixels \* 0.4).toInt()  
  
 // Cap height to 40% of screen height  
 if (calculatedHeight > maxHeightPx) {  
 calculatedHeight = maxHeightPx  
 }  
  
 val layoutParams = videoView.*layoutParams*  
layoutParams.width = parentWidth  
 layoutParams.height = calculatedHeight  
 videoView.*layoutParams* = layoutParams  
  
 videoView.start()  
 **}**  
}

MediaController aggiunge i controlli tipici dei Video Player sopra il video (pausa, play, ...).

Viene poi impostato l’uri del video e preparato un listener per quando il MediaPlayer sarà pronto.

Una volta pronto e controllato che il video abbia dimensioni valide, le dimensioni di VideoView vengono ridimensionate per impostare il corretto aspect ratio al video mettendo come limite di altezza un quarto dello schermo.

Infine, il video viene avviato.

Le seguenti funzioni vengono richiamate dal Controller per eseguire le azioni riconosciute dalla Camera:

fun togglePlayPause() {  
 if (videoView.*isPlaying*) {  
 videoView.pause()  
 } else {  
 videoView.start()  
 }  
}  
  
fun seekForward(ms: Int = 10\_000) {  
 val newTime = videoView.*currentPosition* + ms  
 videoView.seekTo(newTime.*coerceAtMost*(videoView.*duration*))  
}  
  
fun seekBackward(ms: Int = 10\_000) {  
 val newTime = videoView.*currentPosition* - ms  
 videoView.seekTo(newTime.*coerceAtLeast*(0))  
}

La prima mette il video in pausa o riprende la riproduzione del video a seconda dello stato corrente del video.

Le altre due funzioni aggiungono o tolgono un certo numero di millisecondi (di default 10 secondi) al video in riproduzione rispettivamente.

#### VolumeManager

È la classe per la gestione del volume del dispositivo.

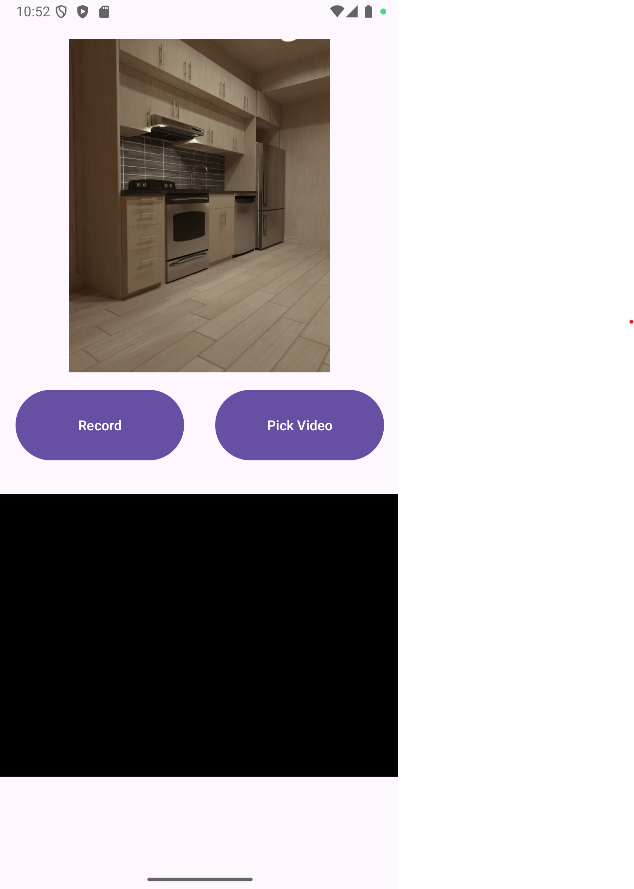
E contiene le seguenti funzioni che verranno richiamate dal Controller se viene riconosciuta la corrispondente azione:

class VolumeManager(private val context: Context) {  
  
 private val audioManager = context.getSystemService(Context.*AUDIO\_SERVICE*) as AudioManager  
  
 fun raiseVolume() {  
 audioManager.adjustStreamVolume(  
 AudioManager.*STREAM\_MUSIC*,  
 AudioManager.*ADJUST\_RAISE*,  
 AudioManager.*FLAG\_SHOW\_UI*  
)  
 }  
  
 fun lowerVolume() {  
 audioManager.adjustStreamVolume(  
 AudioManager.*STREAM\_MUSIC*,  
 AudioManager.*ADJUST\_LOWER*,  
 AudioManager.*FLAG\_SHOW\_UI*  
)  
 }  
}

Queste funzioni rispettivamente aumentano o diminuiscono il volume del dispositivo di una unità mostrando l’ui del volume del dispositivo a schermo attraverso l’utilizzo del servizio di gestione dell’audio di sistema.

### View

Questo rappresenta il modo in cui l’applicazione si presenta all’utente ed è visualizzato in questo modo:



Definito nel seguente file xml:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout 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:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
  
 <LinearLayout  
 android:id="@+id/linearLayout"  
 android:layout\_width="411dp"  
 android:layout\_height="127dp"  
 android:layout\_marginTop="340dp"  
 android:gravity="center\_horizontal"  
 android:orientation="horizontal"  
 android:padding="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/previewView">  
  
 <Button  
 android:id="@+id/recordButton"  
 android:layout\_width="0dp"  
 android:layout\_height="80dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_weight="1"  
 android:onClick="startOrStopRecording"  
 android:text="Record" />  
  
 <Button  
 android:id="@+id/pickVideoButton"  
 android:layout\_width="0dp"  
 android:layout\_height="80dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_weight="1"  
 android:onClick="onPickVideo"  
 android:text="Pick Video" />  
  
 </LinearLayout>  
  
 <androidx.camera.view.PreviewView  
 android:id="@+id/previewView"  
 android:layout\_width="268dp"  
 android:layout\_height="342dp"  
 android:layout\_margin="16dp"  
 android:background="#000000"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.496"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <FrameLayout  
 android:id="@+id/videoContainer"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:paddingBottom="16dp"  
 app:layout\_constraintTop\_toBottomOf="@+id/linearLayout"> <!-- Add padding here -->  
  
 <VideoView  
 android:id="@+id/videoView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="290dp"  
 android:layout\_gravity="center"/>  
 </FrameLayout>  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>