# BAB III
# METODE KERJA

## 3.1 Waktu dan Tempat

Untuk waktu dan tempat pelaksanaan program kegiatan MBKM, yaitu dilakukan pada rentang waktu 21 Februari 2022 – 22 Juli 2022 dengan pelaksanaan dilakukan secara daring, sehingga untuk tempat pelaksanaan lebih fleksibel.

## 3.2 Alat dan Bahan

Untuk alat dan bahan yang kami gunakan untuk membuat *project* yaitu sebagai berikut :

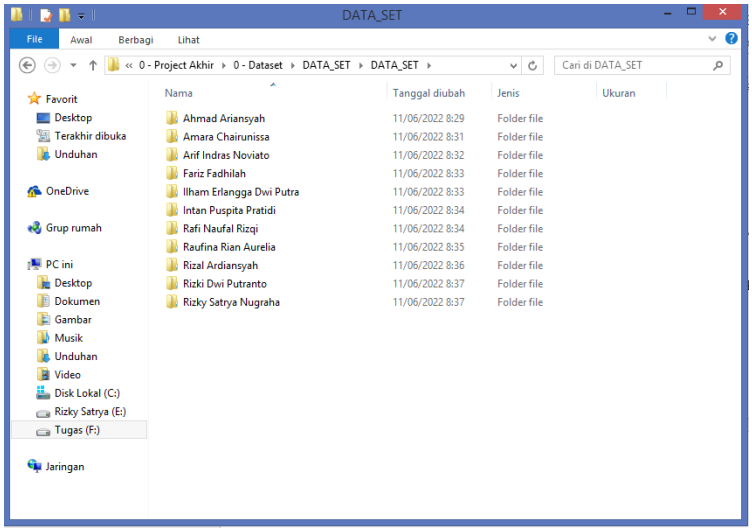
1. Dataset berupa gambar KTM milik mahasiswa IT Telkom Purwokerto
2. Komputer atau laptop
3. Aplikasi Android Studio & Google Colab

## 3.3 Metode dan Proses Kerja

Dalam metode dan proses kerja, langkah pertama yang dilakukan adalah dengan mencari dataset. Dataset yang digunakan adalah gambar KTM dari beberapa mahasiswa yang ada di IT Telkom Purwokerto. Sebagai contoh, seperti gambar berikut ini.

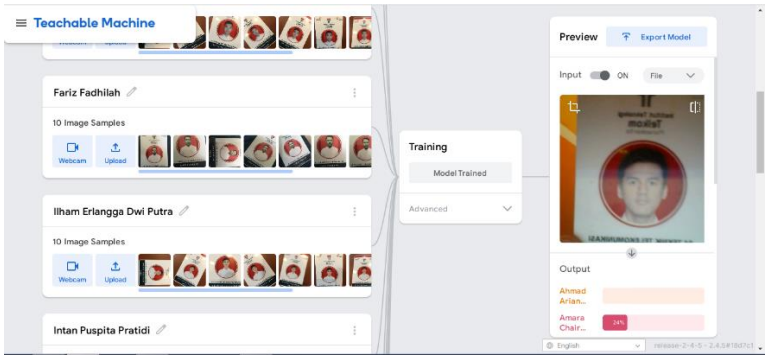

Gambar 3.3.1 Contoh dataset dengan KTM Rizky Satrya

Dengan dataset tersebut, dipilah sesuai dengan pemilik dari KTM tersebut. Setiap KTM, difoto sebanyak 10 gambar, dan dimasukkan ke dalam folder yang sama. Dalam satu folder dataset tersapat 11 kelas (11 nama pemilik KTM) yang masing-masing memiliki 10 gambar di dalamnya.




Gambar 3.3.2 Contoh kelengkapan dataset

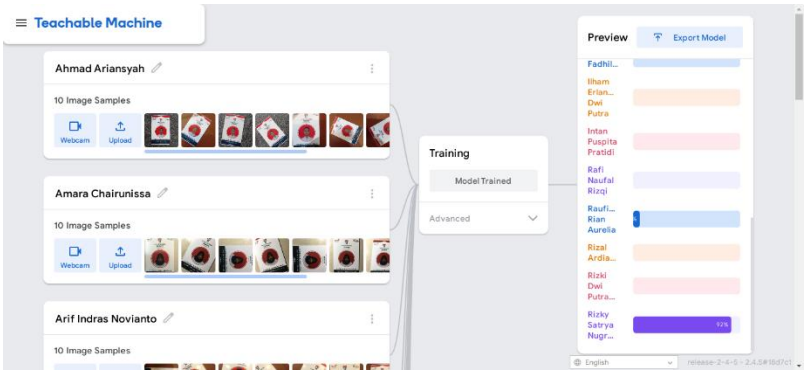
Dataset tersebut dapat diolah menjadi sebuah sistem pendeteksi objek dengan dua cara. Cara pertama, yaitu menggunakan sebuah aplikasi besutan Google, yaitu *Teachable Machine*. Pada aplikasi ini, secara otomatis akan mempelajari sebuah gambar atau *file* yang dimasukkan oleh pengguna. Cara memasukkan dataset juga tergolong mudah, karena hanya dengan *upload* gambar dari dataset, aplikasi tersebutpun langsung bisa melatih dataset yang diberikan tersebut. Ketika semua dataset sudah dimasukkan dan dilatih untuk pembuatan model, kemudian pada aplikasi akan bisa mendeteksi objek dengan dua cara juga. Yaitu melalui kamera, atau *file* gambar yang akan dideteksi. Jika, memilih dengan *file* gambar, terdapat catatan tersendiri, yaitu gambar yang digunakan haruslah *file* yang terdapat gambar KTM dari 11 nama yang ada di dataset.




Gambar 3.3.3 Contoh deteksi objek menggunakan kamera

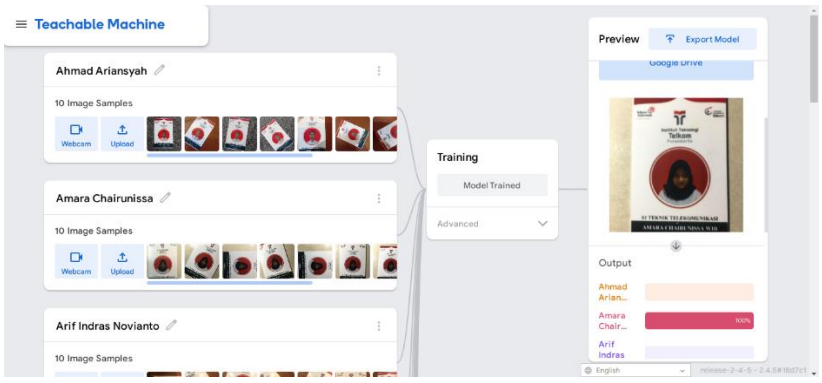
Seperti terlihat pada gambar 3.3.3, bahwa dataset tersebut sudah dilatih dan dapat digunakan untuk pendeteksian objek. Pada gambar tersebut digunakan cara dengan kamera. Jadi, pengguna harus mendekatkan KTM ke arah kamera, pada

aplikasi juga terdapat seberapa keyakinan sistem dalam mendeteksi gambar tersebut, di situ tertampil bahwa sistem memiliki keyakinan bahwa gambar tersebut adalah “Amara Chairunissa” sebanyak 24%, namun jika dilihat lagi, akan tertampil keyakinan sistem bahwa gambar tersebut adalah “Rizky Satrya Nugraha” sebanyak 92%, yang berarti memang sistem sudah mengenali betul bahwa itu adalah “Rizky Satrya Nugraha”.



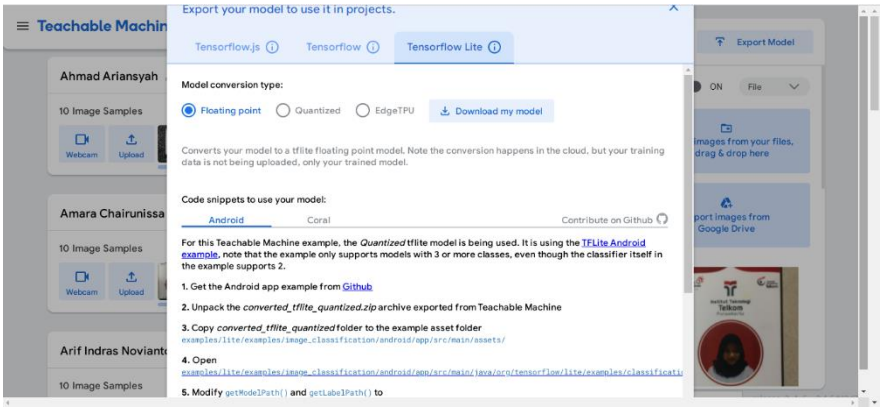

Gambar 3.3.4 Jumlah keyakinan system terhadap gambar

Tapi, ada cara kedua yaitu dengan memasukkan gambar KTM ke sistem. Sebagai contoh, terdapat pada gambar berikut.



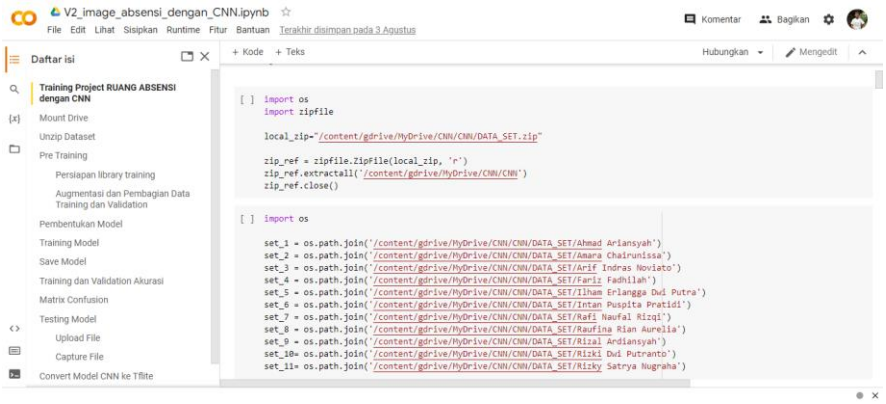

Gambar 3.3.5 Contoh deteksi objek menggunakan *file*

Pada gambar 3.3.5, terdapat contoh jika pendeteksian objek menggunakan *file*. Pada contoh menggunakan gambar dari KTM Amara, dan pada sistem juga meyakini bahwa gambar tersebut adalah “Amara Chairunissa”. Kemudian, pada hasil *training model* yang dilakukan di aplikasi ini, akan di-*export* ke format TF-*Lite* yang nantinya akan digunakan pada aplikasi Android Studio. Kesimpulannya, cara pertama ini menggunakan aplikasi *teachable machine* yang cukup mudah penggunaannya untuk pemula, pengguna hanya diminta untuk memasukkan dataset yang nantinya akan dilatih secara otomatis oleh sistem.



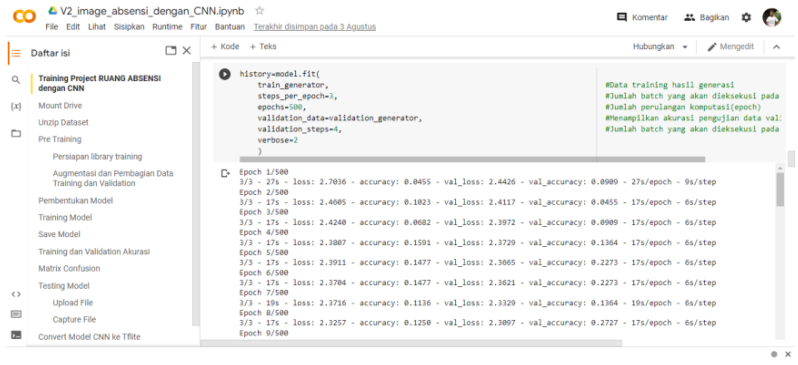

Gambar 3.3.6 Proses *export* model ke TF-*Lite*

Sebelumnya adalah langkah dalam pembuatan model dengan cara menggunakan aplikasi *teachable machine*. Namun, ada satu cara lagi yang bisa dilakukan yaitu dengan aplikasi Google Colab. Cara ini dilakukan dengan memprogram sebuah model dengan bahasa pemrograman Python untuk melatih dataset agar menjadi model yang siap untuk digunakan.



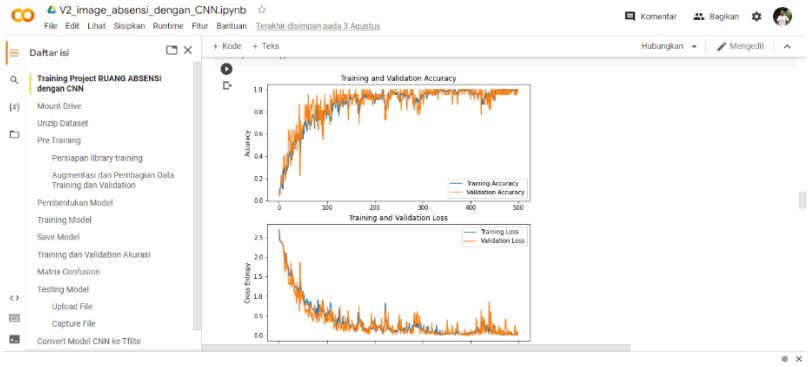

Gambar 3.3.7 Proses *import* dataset ke program

Jika melalui cara ini, langkah pertama yang harus dilakukan adalah dengan memasukkan dataset yang ada ke program Python, cara memasukkannya adalah dengan *import* gambar yang sudah di-*upload* terlebih dahulu di Google Drive. Hal ini dilakukan untuk mempermudah dalam proses memasukkan dataset ke program Python. Langkah selanjutnya, secara ringkas adalah melatih model dengan dataset yang sudah diberikan. Bisa dilakukan dengan melatih model melakukan pendeteksian secara otomatis secara berkala sesuai dengan perintah yang diberikan, untuk percobaan kali ini, diberikan perintah untuk *train model* sebanyak 500 kali, sehingga sistem akan melakukan latihan sebanyak 500 kali secara berulang hingga didapatkan hasil yang lebih akurat. Proses *train* juga cukup lama, sekitar 1 jam.




Gambar 3.3.8 Proses *train model*

Dari hasil latih model tersebut, terlihat bahwa semakin sering model dilatih, tingkat akurasi yang didapatkan semakin baik. Jika akurasi bisa mendekati 1, artinya sistem sudah sangat yakin dengan kemampuan untuk mendeteksi gambar dengan dataset yang diberikan. Terdapat cara juga untuk melihat grafik akurasi, seperti pada gambar berikut.

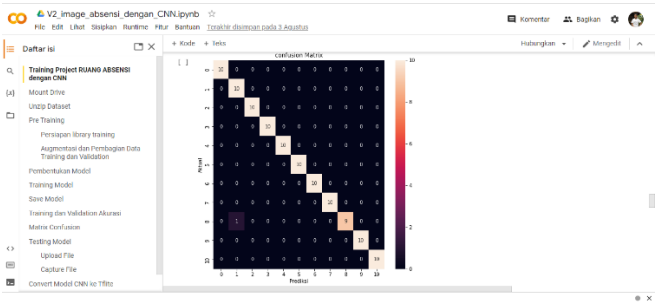



Gambar 3.3.9 Grafik *training and validation accuracy* dan *loss*

Pada gambar 3.3.9 tertampil hasil latihan dan validasi data yang berupa nilai akurasi atau kesalahan yang ada. Pada grafik terlihat bahwa semakin lama model dilatih, maka hasil akurasi yang didapat semakin bagus, dan juga tingkat kesalahan yang terjadi semakin mengecil, yang artinya model semakin baik.

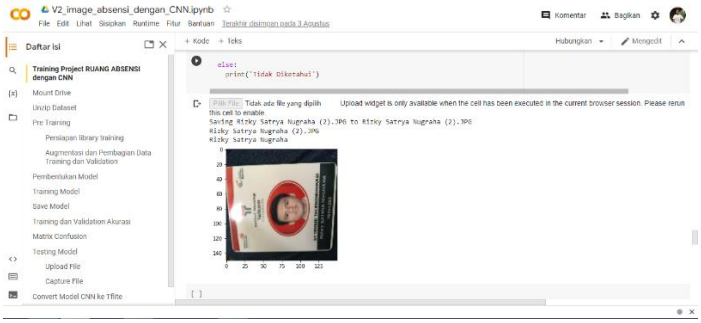
Namun, dari model tersebut juga sempat mengalami kebingungan dalam mendeteksi objek. Ketika dites dengan *confusion matrix*, tertampil hasil bahwa dataset gambar “Rizal Ardiansyah” sedikit membuat kebingungan sistem dalam pendeteksian, menjadi “Amara Chairunissa”, tapi hasil tersebut memang lumrah terjadi, mengingat model yang digunakan adalah CNN. Mungkin dapat lebih baik lagi jika ada pembaruan dalam penggunaan model tersebut. Memang pada

kenyataan dalam proses pendeteksian objek, terutama dengan kamera, akan sedikit keliru.



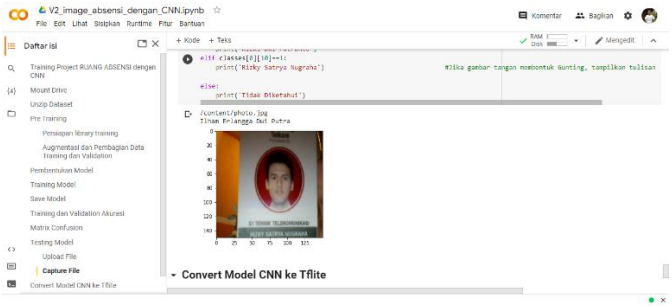
Gambar 3.3.10 *Confusion matrix* yang dihasilkan

Selanjutnya adalah tes hasil dari model yang sudah dibuat, ada dua cara untuk melakukan tes model di sini, yaitu dengan kamera atau dengan *file* gambar yang di-*upload*. Langkah pertama yang dicoba adalah dengan melakukan *upload* gambar pada program.



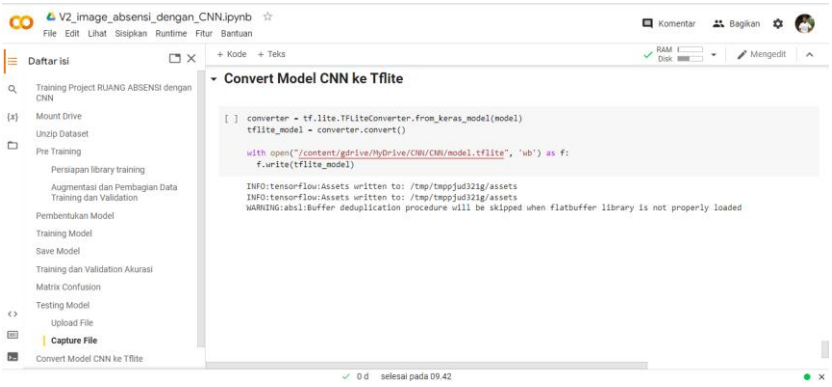
Gambar 3.3.11 Proses tes model dengan *upload* gambar

Pada gambar 3.3.11 adalah contoh dari proses identifikasi gambar dengan cara *upload* gambar. Pada gambar tertampil bahwa gambar yang di-*upload* adalah KTM Rizky Satrya, dan ketika sistem melakukan identifikasi objek gambar tersebut, sistem meyakini bahwa itu adalah "Rizky Satrya Nugraha" dan itu adalah hasil yang benar.



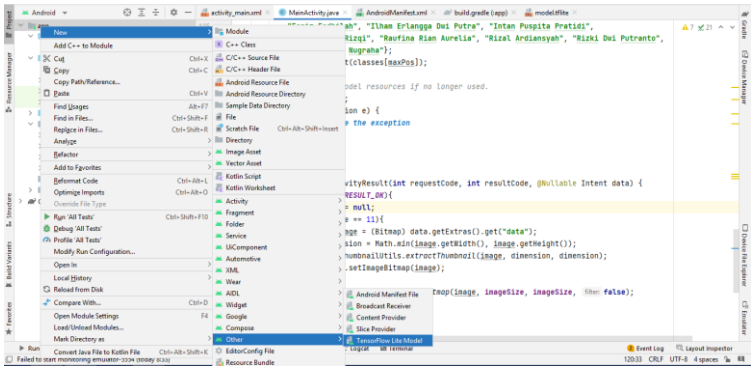
Gambar 3.3.12 Proses tes model dengan kamera

Pada gambar 3.3.12 adalah contoh tes program dengan menggunakan kamera, tertampil pada gambar bahwa ada sedikit kekeliruan, pada KTM yang didekatkan ke kamera adalah KTM Rizky Satrya, namun dideteksi oleh sistem menjadi "Ilham Erlangga", itu bisa terjadi karena mungkin sistem mengira gambar "Rizky Satrya" memiliki kemiripan dengan "Ilham Erlangga". Tapi hal itu bisa dihindari dengan melakukan tes dengan *upload* gambar yang sama dengan gambar dataset yang dilatih. Kemudian, sama seperti sebelumnya saat pelatihan model menggunakan *Teachable Machine*, bahwa model tersebut yang sudah dilatih, perlu di-*export* ke format TF-*Lite*, agar dapat digunakan di Android Studio.



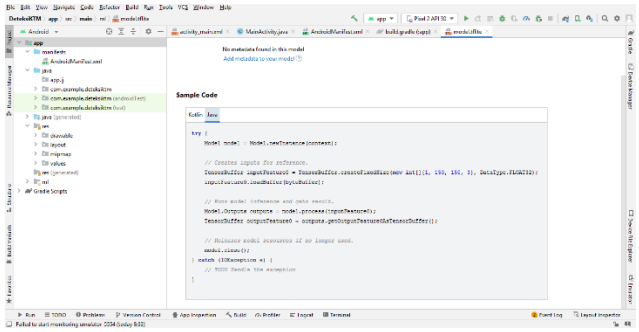

Gambar 3.3.13 Proses *export* model ke TF-*Lite*

Pada saat langkah ini, sudah didapatkan model dengan format TF-*Lite*, dari kedua cara yang digunakan, yaitu dengan *teachable machine* dan juga program Python yang menggunakan Google Colab. Dari model tersebut maka dimasukkan ke Android Studio dengan cara seperti berikut.



Gambar 3.3.14 Proses memasukkan model TF-*Lite*

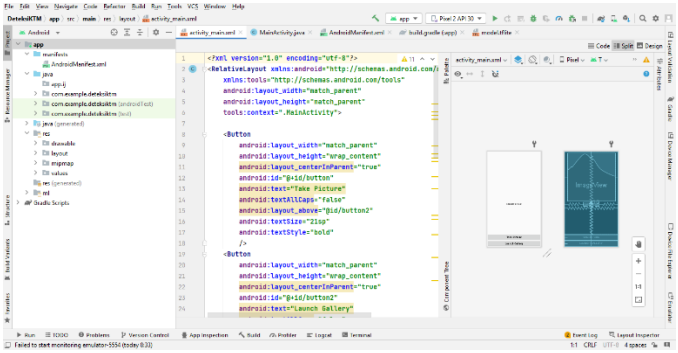
Dari hasil tersebut, maka akan dimasukkan ke folder aplikasi yang sedang dibuat di Android Studio pada folder "ml", model tersebut digunakan pada file *Main Activity* dengan menggunakan bahasa Java.



Gambar 3.3.15 Proses saat sudah *import* model ke Android Studio

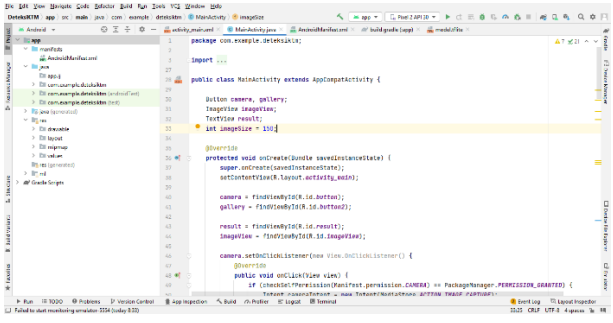
Dari hasil ini, terdapat kode program, yang akan dimasukkan ke *Main Activity*, untuk proses gambar dari dataset yang sebelumnya sudah dimasukkan. Model tersebut adalah cerminan dari hasil *training data* yang sudah dilakukan pada dua cara sebelumnya.

Pada Android Studio juga dibuat desain UI dari aplikasi yang akan digunakan untuk berinteraksi antar model dan juga pengguna aplikasi.



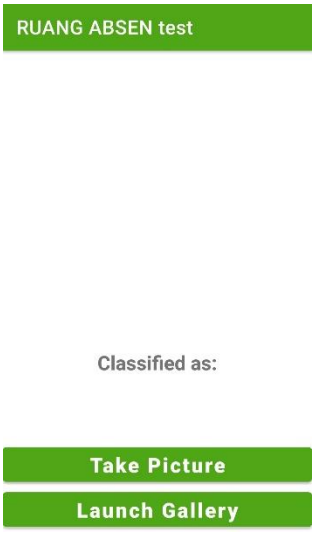
Gambar 3.3.16 Desain UI aplikasi

Desain yang dibuat sangat sederhana, hanya ada pilihan "*Take Picture*" dan "*Import Gallery*" seperti yang ada pada pelatihan model sebelumnya, yang nantinya akan tertampil hasilnya pada kotak *ImageView*, dan terlihat juga hasil klasifikasinya sebagai siapa.



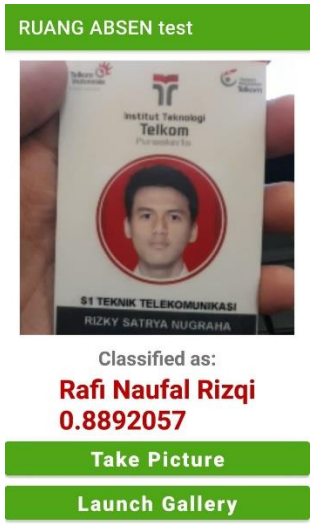
Gambar 3.3.17 Proses sinkronisasi model dengan aplikasi

Pada gambar 3.3.17, tertampil kode program dalam bahasa Java untuk proses sinkronisasi model dengan aplikasi, jadi pada model terdapat gambar yang perlu diketahui oleh aplikasi, agar dapat dimasukkan *file* ketika aplikasi digunakan. Dari sini, aplikasi hanya perlu dijalankan, dan dilakukan tes untuk mengetahui seberapa jauh kemampuan model tersebut, tapi seharusnya hasil yang diberikan tidak jauh berbeda dengan hasil yang ada di program Python maupun *teachable machine*, karena memang perbedaannya ketika dimasukkan ke Android Studio, model tersebut diberikan sebuah 'baju' agar lebih menarik.




Gambar 3.3.18 Tampilan awal aplikasi

Dari tampilan tersebut, memiliki dua fitur yaitu pendeteksian gambar dengan cara ambil gambar langsung atau melalui gambar yang sudah ada di galeri




Gambar 3.3.19 Pendeteksian gambar dengan kamera

Pada gambar 3.3.19 yaitu proses pendeteksian gambar dengan kamera, hal ini dilakukan dengan cara mengambil gambar secara langsung, dan di sini seringkali terdapat kesalahan dalam pendeteksian gambar, hal ini dikarenakan pada model juga masih sering ditemukan kesalahan yang sering dilakukan jika memilih pendeteksian gambar dengan cara ini, seperti tertampil pada gambar bahwa KTM yang diberikan adalah milik "Rizky Satrya" namun pada aplikasi justru dideteksi sebagai "Rafi Naufal Rizqi" dengan tingkat keyakinan 89%, walaupun sebetulnya bahwa ini cukup keliru.

Namun, berbeda halnya dengan cara pendeteksian gambar melalui *file*, apabila gambar yang diberikan adalah gambar yang sama seperti yang ada pada dataset, maka kemungkinan besar hasilnya akan benar.


Gambar 3.3.20 Pendeteksian gambar melalui *file* yang ada di galeri

Seperti tertampil pada gambar, bahwa *file* yang diberikan merupakan gambar yang digunakan juga di dataset, maka kemungkinan besar hasilnya akan berhasil, karena sistem sudah mengenali gambar tersebut sesuai dengan kelas yang ada pada dataset. Pada gambar 3.3.20 juga terlihat bahwa aplikasi meyakini bahwa KTM yang dimaksud adalah "Rizky Satrya" dengan tingkat keyakinan mencapai 96%, dan hasil ini memang benar. Kesimpulannya, model ini bisa digunakan hanya untuk sarana pembelajaran, jika digunakan untuk tingkat yang lebih lanjut, dianjurkan untuk menggunakan model yang lebih bagus seperti YOLO (*You Only Look Once*) dan semacamnya.