BAB II
PROSEDUR KERJA

2.1 Deskripsi Penugasan Kerja

1. *Front End*

   Pekerjaan :

   Mendesain UI dari aplikasi yang hendak dibuat menggunakan android studio dan *deployment* dari model CNN ke aplikasi.

   Pengalaman/keterampilan yang diperoleh :

   Dapat mengetahui dasar-dasar pembuatan aplikasi sederhana berbasis Android, khususnya dalam hal desain aplikasi.

2. *Back End* (Modelling CNN)

   Pekerjaan :

   Membuat modelling dari CNN menggunakan bahasa Python yang nantinya di-*convert* ke format TF-Lite agar bisa digunakan pada aplikasi.

   Pengalaman/keterampilan yang diperoleh :

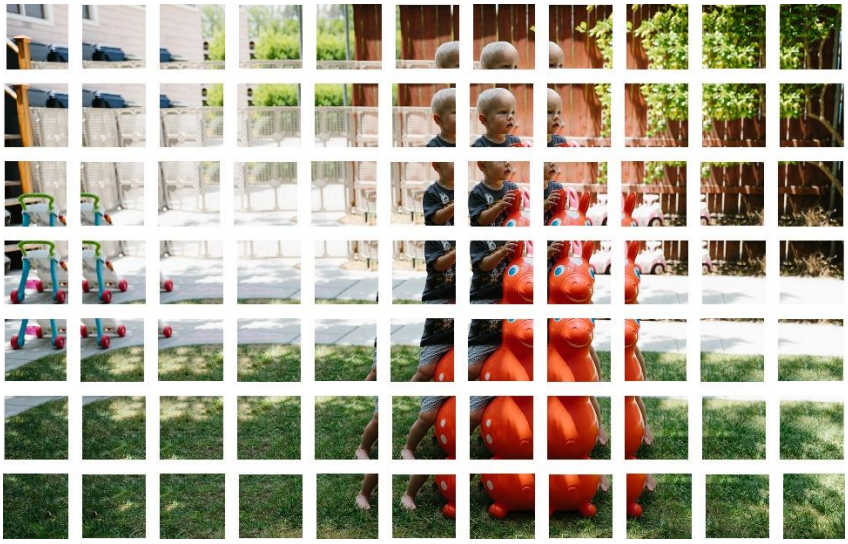
   Dapat mengetahui membuat model CNN, dalam membuat model ini, diperlukan dengan proses training dataset. Jadi, selain pengumpulan dataset, nanti dataset tersebut akan diolah data tersebut agar dapat digunakan untuk memprediksi gambar.

2.2 Teori Dasar Pendukung

CNN (*Convolutional Neural Network*) adalah kategori *Neural Network* yang telah terbukti sangat efektif di berbagai bidang seperti pengenalan dan klasifikasi gambar. CNN juga bisa digunakan untuk mengidentifikasi wajah, objek dan rambu lalu lintas. Untuk tingkat lebih lanjut juga dapat digunakan sebagai robot *vision* dan *self driving car* [1].

Secara garis besarnya, CNN memanfaatkan proses konvolusi dengan menggerakan sebuah kernel konvolusi (filter) berukuran tertentu ke sebuah gambar, computer mendapatkan informasi representative baru dari hasil perkalian bagian gambar tersebut dengan filter yang digunakan [2].

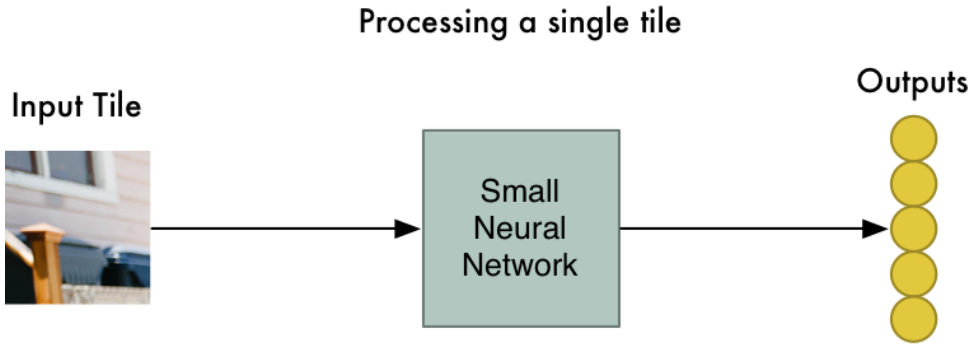
Untuk cara kerjanya sendiri, Langkah pertama yaitu memecah gambar menjadi gambar yang lebih kecil yang tumpeng tindih [2].



Gambar 2.2.1 Pemecahan gambar menjadi 77 bagian [2]

Gambar di atas menunjukan hasil dari gambar yang sudah dipecah menjadi 77 gambar, dengan konvolusi yang sama. Hal ini bertujuan untuk mempermudah dalam pembuatan model [2].

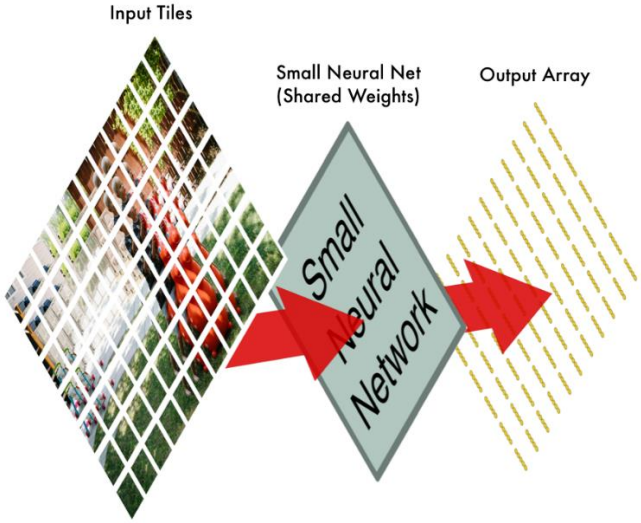
Langkah kedua yaitu memasukkan setiap gambar yang lebih kecil ke *small neural network*, dari gambar 2.2.1 di atas, setiap gambar kecil dari hasil konvolusi tersebut kemudian dijadikan *input* untuk menghasilkan sebuah representasi fitur. Hal ini memberikan CNN kemampuan mengenali sebuah objek, dimanapun posisi objek tersebut muncul pada sebuah gambar [2].



Gambar 2.2.2 Proses gambar hasil konvolusi [2]

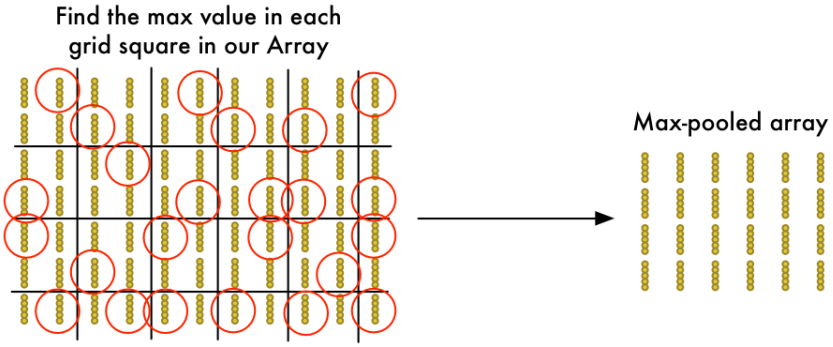
Proses ini dilakukan untuk semua bagian dari masing-masing gambar kecilnya, dengan menggunakan filter yang sama. Dengan kata lain, setiap bagian gambar akan memiliki factor pengali yang sama, atau dalam konteks *neural network* disebut sebagai *weights sharing.* Jika ada sesuatu yang tampak menarik di setiap gambarnya, maka akan ditandai bagian itu sebagai *object of interest* [2].

Langkah ketiga yaitu menyimpan hasil dari masing-masing gambar kecil ke dalam *array* baru. Maka hasilnya akan seperti pada gambar 2.2.3 di bawah.



Gambar 2.2.3 Proses hasil *input* pecahan gambar [2]

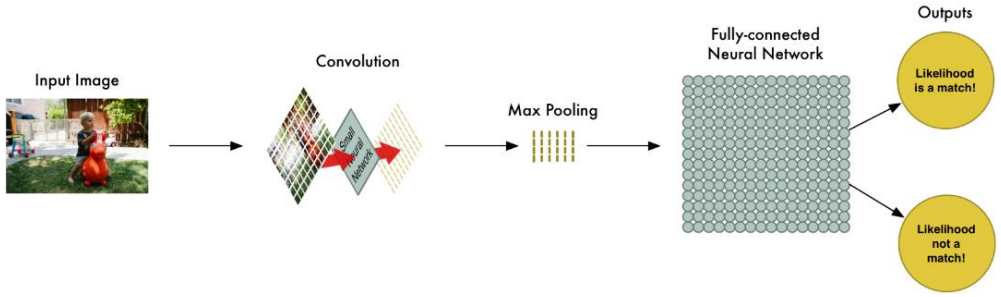
Langkah keempat yaitu *downsampling*, Ketika langkah ketiga, *array* masih terlalu besar, maka untuk mengecilkan ukuran *array*-nya digunakan *downsampling* yang penggunaannya dinamakan *max pooling* atau mengambil nilai *pixel* terbesar di setiap *pooling kernel*. Dengan begitu, sekalipun mengurangi jumlah parameter, informasi terpenting dari bagian tersebut akan tetap diambil [2].



Gambar 2.2.4 Proses *max pooling* [2]

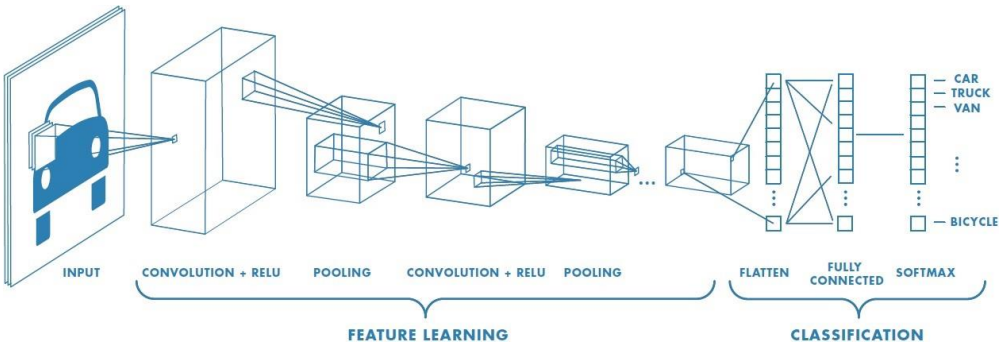
Langkah kelima atau langkah terakhir yaitu dengan membuat prediksi, Ketika gambar yang besar sudah dipecah menjadi *array* yang cukup kecil, maka dari *array* (sekelompok angka) tersebut digunakan untuk di-*input*-kan ke dalam jaringan saraf lain. Jaringan saraf paling terakhir akan memutuskan apakah gambarnya cocok atau tidak. Untuk memberikan perbedaan dari langkah konvolusi, maka bisa kita sebut dengan "*fully connected*" *network* [2].

Secara garis besarnya, langkah-langkah di atas tampak seperti gambar 2.2.5 di bawah ini.



Gambar 2.2.5 Langkah membuat model CNN [2]

Arsitektur dari CNN sendiri dibagi menjadi dua bagian besar, yaitu *Feature Extraction Layer* dan *Fully Connected Layer*.

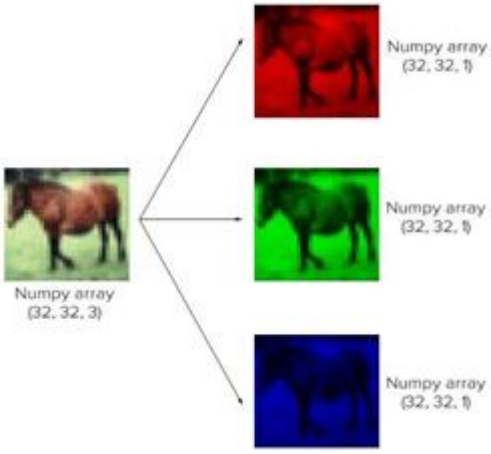


Gambar 2.2.6 Arsitektur CNN [3]

Salah satu arsitektur CNN yang pertama adalah *Feature Extraction Layer*, diberi nama demikian karena ada proses yang terjadi pada bagian ini yaitu "*encoding*" dari sebuah *image* menjadi *features* yang berupa angka-angka yang merepresentasikan *image* tersebut (*feature extraction*). *Feature extraction layer* terdiri dari dua bagian yaitu *Convolutional Layer* dan *Pooling Layer*. Namun kadang ada beberapa riset/*paper* yang tidak menggunakan *pooling* [4].

Convolutional layer terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan Panjang dan tinggi (*pixel*).

Convolutional layer adalah *core* dari *Convolutional Neural Network*. Lapisan ini menghasilkan gambar baru yang berisi *features* dari gambar yang telah di-*input*-kan. Proses yang terjadi pada lapisan ini adalah konvolusi mengaplikasikan filter pada gambar. Filter yang diaplikasikan tersebut adalah berupa matriks bisa ukuran 1×1, 3×3 atau 5×5. Proses konvolusi ini menghasilkan *feature map* yang kemudian akan dipergunakan pada saat lapisan aktivasi [5].



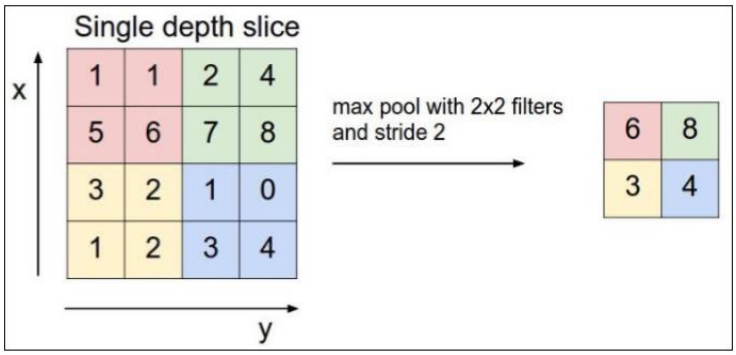
Gambar 2.2.7 Foto yang dibagi menjadi 3 *channel* dengan RGB [5]

Gambar 2.2.7 menunjukkan RGB (*Red, Green, Blue*) yang berukuran 32×32 piksel yang sebenarnya adalah multidimensional *array* dengan ukuran 32×32 piksel (3 adalah jumlah *channel*). Sebagai contoh, *layer* pertama pada *feature extraction layer* adalah *convolutional layer* dengan ukuran 5×5×3. Panjang 5 piksel, tinggi 5 piksel dan tebal/jumlah 3 buah sesuai dengan *channel* dari gambar tersebut.

Ketiga filter ini akan digeser keseluruhan bagian dari gambar. Setiap pergeseranb akan dilakukan operasi "dot" antara *input* dan nilai dari filter tersebut sehingga menghasilkan sebuah *output* atau biasa disebut sebagai *activation map* atau *feature map.*

Setelah *convolutional layer* biasanya terdapat *pooling layer*. Pada prinsipnya *pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* (parameter yang menentukan jumlah pergeseran filter) tertentu yang bergeser pada seluruh area *feature map*. *Pooling* yang biasa digunakan adalah *max pooling* dan *average pooling.* Tujuan dari penggunaan *pooling layer* adalah mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus di-*update* semakin sedikit dan mengatasi *overfitting* [6].

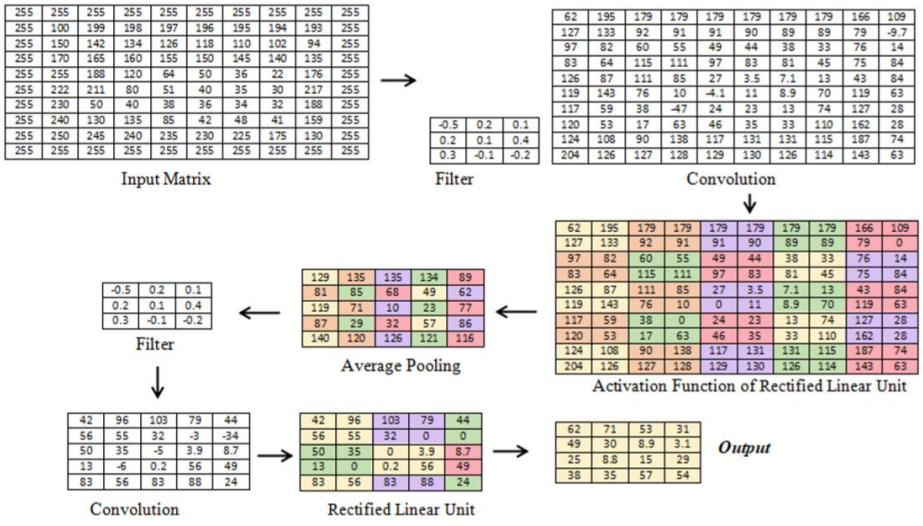
Hal terpenting dalam pembuatan model CNN adalah dengan memilih banyak jenis lapisan *pooling*. Hal ini dapat menguntungkan kinerja model. Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan mengurangi ukurannya. Bentuk lapisan *pooling* yang paling umum adalah dengan menggunakan filter berukuran 2×2 yang diaplikasikan dengan langkah sebanyak 2 dan kemudian beroperasi pada setiap irisan dari *input*. Bentuk seperti ini akan mengurangi *feature map* hingga 75% dari ukuran aslinya [6].



Gambar 2.2.8 Contoh *max pooling* [6]

Lapisan *pooling* akan beroperasi pada setiap irisan kedalaman volume *input* secara bergantian. Pada gambar di atas, lapisan *pooling* menggunakan salah satu operasi maksimal yang merupakan operasi yang paling umum. Gambar 2.2.8 menunjukkan operasi dengan langkah 2 dan ukuran filter 2×2. Dari ukuran *input* 4×4, pada masing-masing 4 angka pada *input* operasi mengambil nilai maksimalnya dan membuat ukuran *output* baru menjadi 2×2 [6].

Lapisan konvolusi yang diaplikasikan untuk mendapatkan *feature map*. Contoh proses konvolusi dengan *input* berupa citra satu *channel* digambarkan seperti pada gambar berikut.



Gambar 2.2.9 Ilustrasi proses konvolusi [6]

Pada gambar 2.2.9, menunjukkan sebuah citra berukuran 10×10 piksel direpresentasikan sebagai matriks. Matriks awal diproses dengan dua *layer* konvolusi untuk mendapatkan *feature map.* Pada *layer* konvolusi pertama, filter yang digunakan berukuran 3×3 dengan bobot yang telah ditentukan . hasil dari konvolusi pertama berupa matriks dengan ukuran 9×9 [6].

Setelah melalui proses konvolusi, fungsi aktivasi dikenakan pada hasil konvolusi. Fungsi aktivasi yang digunakan adalah reLu. *Output* dari fungsi reLu kemudian dikenakan *pooling* dengan filter berukuran 2×2 dan *stride* sebesar dua. Sebelum melakukan *pooling*, dapat digunakan *zero padding* sehingga matriks hasil *pooling* berukuran 5×5. Matriks ini kemudian melalui tahap konvolusi kedua dengan ukuran filter sama seperti sebelumnya, tetapi dengan bobot yang berbeda. Dalam hal ini, ukuran tidak harus sama dengan konvolusi tahap pertama dan merupakan parameter yang bisa dioptimalkan. Sementara bobot matriks merupakan nilai yang dcari melalui proses pembelajaran [7].

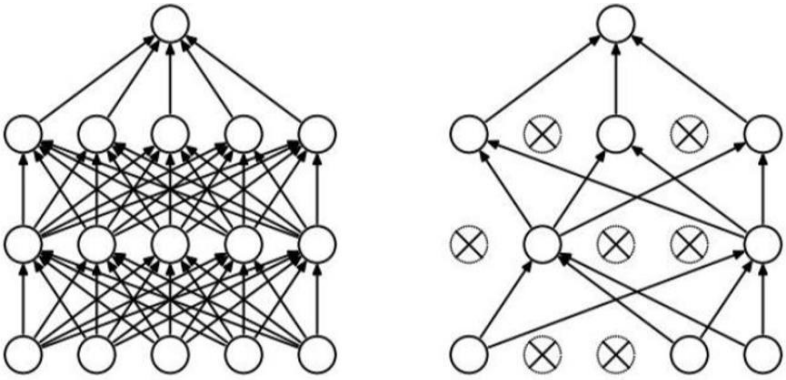
Output dari proses konvolusi tahap kedua dikenakan dengan fungsi aktivasi yang sama, yaitu reLu. *Pooling* yang dikenakan berukuran 2×2 dengan *stride* satu, sehingga menghasilkan matriks dengan ukuran 4×4. Proses konvolusi bisa dilanjutkan sesuai dengan matriks akhir yang diinginkan. Dalam hal ini, jika konvolusi dihentikan sampai tahap kedua, maka matriks berukuran 4×4 tersebut menjadi *input* bagi *neural network*. Jika filter yang digunakan sejumlah n, maka *input* bagi *neural network* adalah n×4×4 *nodes*. Pada praktiknya, penggunaan fungsi aktivasi dan *pooling* bisa dibalik urutannya tanpa mengubah hasil dari konvolusi. Pembalikan ukuran ini bertujuan untuk mengurangi proses yang digunakan sehingga menjadi lebih cepat [7].

Kemudian, untuk arsitektur CNN yang kedua adalah *fully-connected layer* (FC *Layer*). Lapisan *Fully-connected* adalah lapisan dimana semua *neuron* aktivitas dari lapisan sebelumnya terhubung semua dengan *neuron* di lapisan selanjutnya seperti halnya jaringan syaraf tiruan biasa. Setiap aktivitas dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua *neuron* di lapisan *Fully-Connected* [7].

Lapisan *Fully-Connected* biasanya digunakan pada metode multi lapisan *perceptron* dan bertujuan untuk mengolah data sehingga bisa diklasifikasikan. Perbedaan anatar lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah *neuron* di lapisan konvolusi terhubung hanya ke daerah tertentu pada *input*. Sementara lapisan *Fully-Connected* memiliki *neuron* yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoperasikan produk dot, sehinga fungsinya tidak begitu berbeda [7].

Ada juga teknik regularisasi jaringan syaraf yaitu *dropout* yang di mana beberapa *neuron* akan dipilih secara acak dan tidak dipakai selama pelatihan model. *Neuron-neuron* ini dapat dibilang dibuang secara acak. Hal ini berarti bahwa kontribusi *neuron* yang dibuang akan diberhentikan sementara jaringan dan bobot baru juga tidak diterapkan pada *neuron* pada saat *backpropagation* [8].

Dropout merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses *learning*. *Dropout* mengacu kepada menghilangkan neuron yang berupa *hidden* mapun *layer* yang *visible* di dalam jaringan. Dengan menghilangkan suatu *neuron*, berarti menghilangkannya sementara dari jaringan yang ada. *Neuron* yang akan dihilangkan akan dipilih secara acak. Setiap *neuron* akan diberikan probabilitas yang bernilai antara 0 dan 1 [9].



(a) Jaringan Syaraf Biasa

(b) Jaringan Syaraf Setelah dikenakan Teknik *Dropout*

Gambar 2.2.10 Contoh implementasi *dropout regularization* [10]

Pada gambar 2.2.10 di atas jaringan syaraf (a) merupakan jaringan syaraf biasa dengan 2 lapisan tersembunyi. Sedangkan pada bagian (b) jaringan syaraf sudah diaplikasikan teknik regularisasi *dropout* di mana ada beberapa *neuron* aktivasi yang tidak dipakai lagi. Teknik ini sangat mudah diimplementasikan pada model CNN dan akan berdampak pada performa model dalam melatih serta mengurangi *overfitting*.