BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Sebelumnya

Penelitian yang telah dilakukan mengenai sistem untuk mengidentifikasi objek pada wajah telah banyak dilakukan dengan berbagai implementasi. Salah satu implementasi yang biasa digunakan adalah penerapan *Deep Learning* berbasis *Tensorflow* dan *Convolutional Neural Network.* Berikut beberapa penelitian terkait implementasi tersebut ada pada tabel 2.1 :

Tabel 2.1 Referensi Penelitian Sebelumnya

Penelitian Sebelumnya					
No.	**Judul Penelitian**	**Tujuan Penelitian**	**Metode**	**Hasil Penelitian**	**Keterangan**
1	Rancang Bangun New Normal Covid-19 Masker Detektor Dengan Notifikasi Telegram Berbasis *Internet Of Things*,2020	Mengimplementasikan sistem informasi otomatis untuk mendeteksi masker pada wajah berbasis iot.	*haar cascade classifier*	Sistem dapat mendeteksi orang yang menggunakan masker dan notifikasi dapat dikirimkan ke keamanan melalui aplikasi telegram dengan baik	Pada penelitian terebut terdapat kelemahan pada streaming video yang sedikit lambat karena raspberry pi memiliki memory yang terbatas, dimana hanya dapat menjalankan streaming video 20-32 fps, sehingga gambar terlihat sedikit tersendat.

Penelitian sebelumnya					
No.	**Judul penelitian**	**Tujuan penelitian**	**Metode**	**Hasil penelitian**	**Keterangan**
2	Penerapan *face recognition* pada aplikasi akademik online,2020	Membuat sebuah perancangan sistem login akademik online di Universitas Nasional.	*Local Binary Pattern Histogram* dan *metode Haar Cascade Classifier*	hasil percobaan yang telah dilakukan, bahwa penerapan dari ke dua metode *Local Binary Pattern Histogram* dan *metode Haar Cascade Classifier* berhasil untuk mendeteksi wajah mahasiswa secara *realtime.*	Pada penelitian terebut terdapat penerapan dari ke dua metode *Local Binary Pattern Histogram* dan *metode Haar Cascade Classifier* digabungkan dapat digunakan untuk mendeteksi dan mengenali wajah secara *real time* untuk kebutuhan login ke akademik online, dengan kondisi pencahayaan yang cukup dan wajah tidak terhalang oleh benda lain.
3	*Deep learning object detection* pada video menggunakan *tensorflow* dan *convolutional neural network*,2018	Mengetahui motif ukiran jepara menggunakan algoritma *cnn*	*Convolutional neural network*	Tingkat akurasi model yang didapatkan dari hasil pendeteksian klasifikasi citra meja dan kursi motif ukiran jepara pada suatu citra digital menggunakan *convolutional neural network* berkisar antara 70% hingga 99%. 6.2	Pada penelitian tersebut perlu ditambahkan dataset untuk meningkatkan akurasi dan juga perlu menggunakan perangkat yang memiliki spefikasi lebih tinggi yaitu menggunakan komputer dengan random access memory (ram) yang tinggi dan menggunakan graphic processing unit (gpu) untuk mempercepat *training*

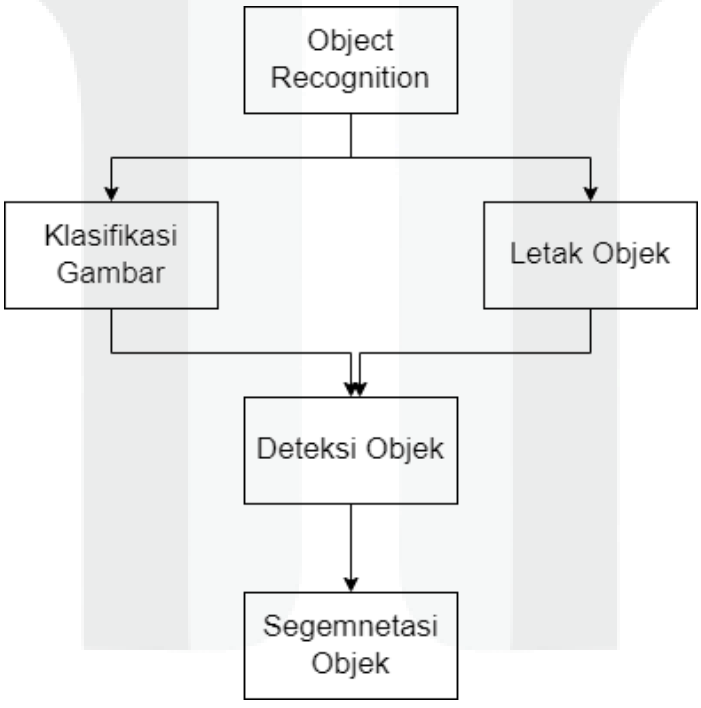
Penelitian sebelumnya					
No.	**Judul penelitian**	**Tujuan penelitian**	**Metode**	**Hasil penelitian**	**Keterangan**
4	*Deep learning* untuk deteksi wajah yang berhijab mengguna kan algoritma *convolutional neural network (cnn)* dengan *tensorflow*,2020	Meningkatakan akurasi pengenalan wajah yang menggunakan hijab menggunakan metode *cnn* pada *framework tensorflow*	*Convolutional neural network*	Hasil implementasi *cnn* menggunakan *tensorflow* dengan menggunakan dataset 300 jumlah gambar wajah yang berhijab mendapat akurasi 92% pada proses *training* dan 87% pada proses testing. Sehingga akurasi dari penelitian ini bisa dikatakan berjalan dengan optimal dalam mendeteksi wajah yang memakai hijab	Pada penelitian tersebut dapat ditambahkan jumlah dataset untuk meningkatkan akurasi pada penerapan model *cnn* dan juga dapat menambahkan objek gambar dengan memakai atribut pada sekitar area wajah yang berbeda dan juga menggunakan lebih banyak ekspresi.

Penelitian sebelumnya					
No.	**Judul penelitian**	**Tujuan penelitian**	**Metode**	**Hasil penelitian**	**Keterangan**
5	Deteksi ekspresi wajah mengguna kan *tensorflow*,2021	Mendeteksi ekspresi wajah menggunakan sistem dengan akurasi yang optimal.	*Convolutional neural network*	Dengan menggunakan *convolutional neural network* untuk mendeteksi ekspresi wajah mendapat hasil akurasi *training* data 62,24%, validasi 62,44%, *training loss* 4,54%, validation loss. Data tersebut diambil dari pendeteksi wajah dengan video secara realtime	Pada penelitian ini dapat disimpulkan bahawa sistem tidak mengalami overfit karena nilai akurasi validasi lebih besar dari nilai akurasi *training*. Parameter optimal tersebut diujikan kepada data asing yang belum pernah di deteksi sebelumnya dan menghasilkan nilai akurasi deteksi 67%, presisi 67% dan *recall* 66%
6	*Retinafacemask: a face mask detector,2020*	Meningkatk an akurasi dan efisiensi pendeteksia n masker menggunak an retinafacem ask	*Convolutional neural network*	Hasil dari penelitian menunjukkan bahwa RetinaFaceMask mencapai hasil mutakhir pada dataset masker wajah publik dengan masing-masing 2,3% dan 1,5% lebih tinggi dari hasil dasar dalam presisi deteksi wajah dan masker, dan 11,0% dan 5,9% lebih tinggi dari baseline untuk mengingat.	Pada penelitian tersebut tulang punggung yang kuat *ResNet* dan tulang punggung ringan *MobileNet* dapat digunakan untuk skenario komputasi tinggi dan rendah, masing-masing. Untuk mengekstrak fitur yang lebih kuat, menggunakan pembelajaran transfer untuk mengadopsi bobot dari deteksi wajah tugas serupa, yang dilatih pada kumpulan data yang sangat besar.

2.2 Dasar Teori

2.2.1 *Konsep Object Recognition*

Object Recognition merupakan salah satu tantangan dalam pembuatan sistem cerdas. Beberapa sistem telah dibuat untuk mengklasifikasikan dan mengenali sebuah gambar. Salah satu algoritma yang menunjukan hasil yang baik pada sistem ini adalah *"Deep neural network"*[12]. Untuk menggunakan algoritma tersebut memerlukan pendekatan yang tidak asing atau popular untuk mengenali objek. Untuk melakukan hal tersebut *developer* harus memiliki ilmu pengetahuan yang mendalam mengenai data yang akan diKelola. Dalam pengelolaan sistem cerdas agar dapat mengenali objek memerlukan dataset yang sesuai dengan objek yang akan dikenali. Dalam proses pengenalan objek *neural network* merupakan salah satu algoritma yang sangat direkomendasikan karena algoritma telah menjadi area aktif peneliti untuk membuat terobosan beberapa tahun terakhir[13]. Cara kerja *Object Recognition* dapat dilihat pada gambar berikut

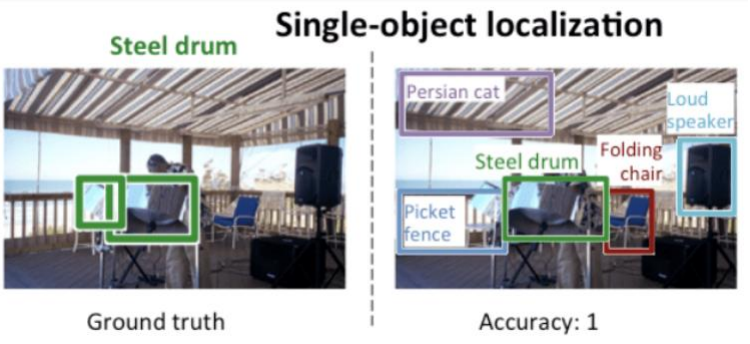


Gambar 2. 1 Cara Kerja *Object Recognition* [12]

Pada gambar diatas menjelaskan mengenai Langkah – Langkah kerja dari *Object Recognition* yang berisi :

1. Klasifikasi gambar : pada tahap ini terisi algoritma yang menghasilkan daftar kategori objek yang ada dalam gambar yang akan dideteksi.
2. Letak objek : pada tahap tersebut menghasilkan daftar objek yang ada pada gambar yang di deteksi, bersama dengan kotak pembatas sejajar dengan sumbu yang akan menunjukan posisi dan skala satu instance dari setiap kategori objek
3. Deteksi objek : berisi algoritma yang menghasilkan daftar kategori objek yang ada pada gambar bersama dengan kotak pembatas sejajar dengan sumbu yang menunjukan posisi dan skala setiap instance dari setiap kategori objek.
4. Segmentasi object : pada tahap ini objek yang telah dideteksi dan di kelompokkan akan dipisahkan dengan objek yang berbeda lainnya.

Pada objek recognition juga terdapat *single object detection* dan *object localization*. Untuk kesederhanaan penggunaan lebih sederhanan *single object* detection dari pada *object localization* karena definisi objek pada *object localization* lebih luas. Berikut gambar mengenai dua tipe *Object Recognition* tersebut . [12]



Gambar 2. 2 Perbedaan dua tipe *object recognition* [12]

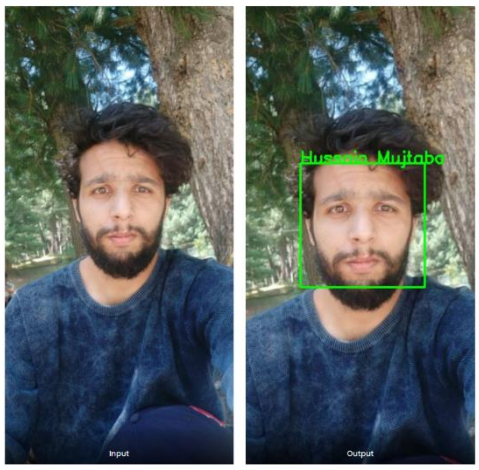
2.2.2 Open Cv dan Python

Open Cv (*Open Source Computer Vision*) adalah pustaka visi komputer populer yang dimulai oleh Intel pada 1999. Pustaka lintas platform menetapkan fokusnya pada pemrosesan gambar waktu nyata dan mencakup implementasi bebas paten dari algoritma visi komputer terbaru. Pada tahun 2008 Willow Garage

mengambil alih dukungan dan *Open Cv* 2.3.1 kini hadir dengan antarmuka pemrograman untuk C, C++, *Python* dan Android . *Open Cv* dirilis di bawah lisensi BSD sehingga digunakan dalam proyek akademik dan produk komersial[14]. *Open Computer Vision* merupakan salah satu *library open source* yang bertujuan untuk melakukan pengolahan citra agar dapat mempunyai kesamaan fungsi seperti pengolahan visual pada manusia [15]. *Open Cv* memiliki banyak bahasa pemrograman yang mendukung dan dapat digunakan. *Open Cv* dapat digunakan untuk mengidentifikasi suatu objek, wajah, gerakan hingga gestur[16].

Python merupakan salah satu bahasa pemrograman yang bersifat interpreter, *interactive, object-oriented* dan dapat beroperasi hampir di semua *platform* seperti windows, mac dan linux[15]. *Python* umumnya digunakan dalam proyek kecerdasan buatan dan proyek pembelajaran mesin dengan bantuan perpustakaan seperti *Tensorflow , Keras , Pytorch* dan *Scikit-learn* [17][18][19][20].

Berdasarkan penelitian yang dibuat oleh husain Mujtaba dalam artikel *great Learning* menjelaskan mengenai implementasi pengenalan wajah dengan *Python* dan *Open Cv*. Dalam penelitian tersebut *Open Cv* sebagai perpustakaan untuk memproses sebuah gambar video dan *Python* sebagai bahasa program untuk menjalankan program[21]. Adapun contoh implementasi penelitian tersebut pada gambar 2.3.



Gambar 2. 3 Implementasi *opencv* dan *python* [21]

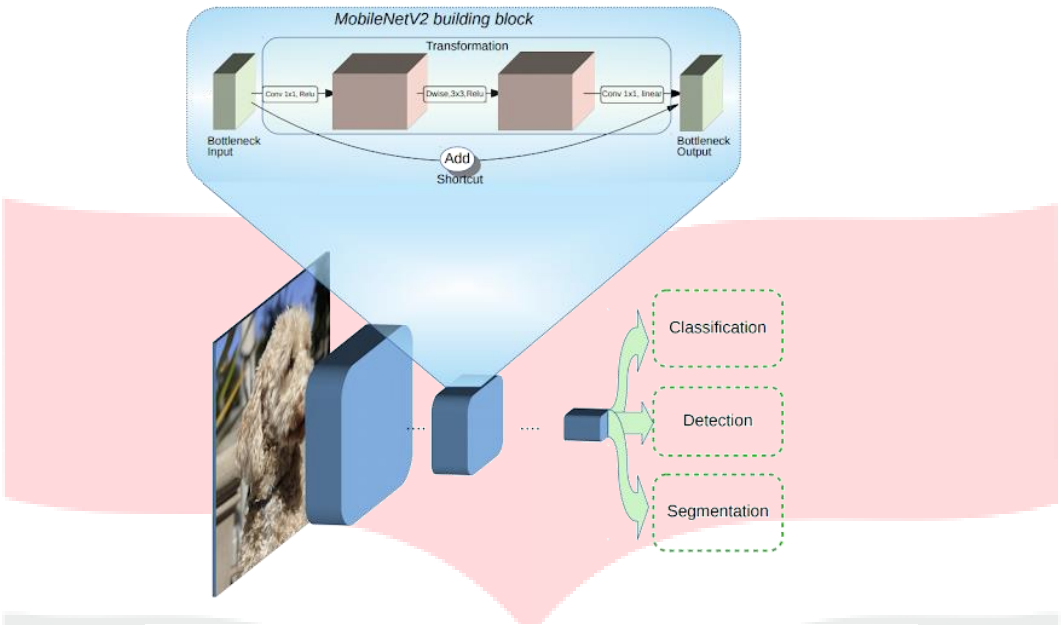
Pada gambar 2.3 menjelaskan hasil dari pengolahan pengenalan wajah pada penelitian. Pada gambar di sebelah kiri adalah data yang dimasukan ke dalam program dan gambar sebelah kanan merupakan hasil pengolahan bisa dilihat dari kotak hijau yang memutari wajah.

2.2.3 Tensorflow & Keras

Tensorflow [10] adalah generasi kedua dari sistem penelitian dan pengembangan kecerdasan buatan yang dikembangkan oleh Google, yang mendukung jaringan saraf *Convolutional* (*CNN*) [22]. Tensor adalah unit data pusat dalam *Tensorflow*. Tensor terdiri dari sekumpulan tipe data primitif yang dibentuk menjadi array dari dimensi apa pun. Tipe data primitif tensor seperti bilangan bulat, titik pecahan, karakter, string, dan sebagainya [23]. Dalam penggunaan *Tensorflow* terdapat satu *packages* yang dapat digunakan yaitu keras. Penggunaan keras untuk mendukung kinerja *Tensorflow* sebagai *optimizer* agar mendapatkan hasil yang cepat dan bagus. Jadi *keras* merupakan *wrapper* dari *Tensorflow* untuk lebih mengefisienkan atau memudahkan[24]. Berdasarkan penelitian yang dibuat Xiao-Ling Xia, Cui Xu, Bing Nan menjelaskan pengenalan wajah menggunakan platform *Tensorflow inception* v3, pada penelitian mereka menggunakan transfer *Learning* techniques untuk melatih data yang dapat menjadi akurasi pengenalan wajah . *Tensorflow* mempunyai beberapa modul yaitu :

2.2.3.1 MobileNet v2

MobileNet V2 adalah salah satu arsitektur *Convolutional* neural network (CNN) berbasis ponsel yang dapat digunakan untuk mengatasi kebutuhan akan computing resource berlebih. MobileNet V2 merupakan penyempurnaan dari arsitektur MobileNet. Arsitektur MobileNet dan arsitektur CNN pada umumnya memiliki perbedaan pada penggunaan lapisan atau convolution layer. Convolution layer pada *MobileNetV2* menggunakan ketebalan filter yang sesuai dengan ketebalan dari input image[25]. Berikut gambaran arsitektur *MobileNetV2* pada gambar 2.5.



Gambar 2. 4 Arsitektur *MobileNetV2*

Gambar 2.4 merupakan arsitektur *MobileNetV2*. *MobileNetV2* masih menggunakan depthwise dan pointwise convolution. *MobileNetV2* menambahkan dua fitur baru yaitu: 1) linear bottleneck, dan 2) shortcut connections antar bottlenecks. Struktur dasar dari arsitektur ini ditunjukkan pada gambar 2.4. Pada bagian bottleneck terdapat input dan output antara model sedangkan lapisan atau layer bagian dalam meng-enkapsulasi kemampuan model untuk mengubah input dari konsep tingkat yang lebih rendah ke deskriptor tingkat yang lebih tinggi. Pada akhirnya, seperti halnya koneksi residual pada CNN tradisional, shortcut antar bottlenecks memungkinkan training atau pelatihan yang lebih cepat dan akurasi yang lebih baik.

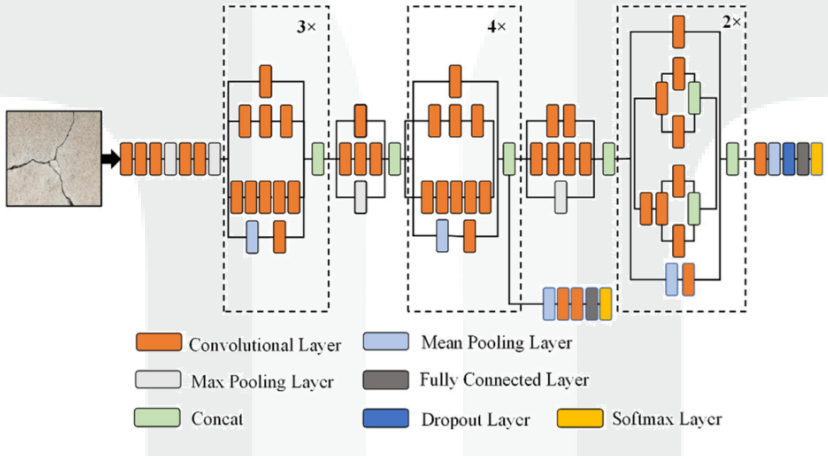
2.2.3.2 *INCEPTIONV3*

Inception-v3 [26] adalah model jaringan Google setelah *Inception*-v1 [27], *Inception-v2* [28] pada tahun 2015. Ini adalah pemikiran ulang untuk struktur awal visi komputer.

Model *Inception-v3* [26] terutama didasarkan pada prinsip-prinsip dasar desain jaringan perubahan struktur *Inception*, metode utamanya adalah:

dekomposisi konvolusi filter ukuran besar, pengklasifikasi tambahan, mengurangi ukuran peta fitur, dll

Jaringan Inception-v3 ialah jaringan yang kompleks, akan memakan biaya setidaknya beberapa hari atau bahkan beberapa hari Jika kita melatih model secara langsung. dengan menggunakan metode transfer Learning, parameter lapisan sebelumnya tidak berubah, namun hanya lapisan terakhir yg dilatih. Lapisan terakhir artinya pengklasifikasi softmax. Pengklasifikasi ini artinya 1000 Node keluaran pada jaringan asli (ImageNet mempunyai 1000 kelas), jadi kita perlu menghapus lapisan jaringan terakhir, serta kemudian melatih pulang lapisan terakhir. Alasan buat melatih ulang lapisan terakhir artinya buat bekerja pada kelas objek baru. Akibatnya, mengidentifikasi 1000 kelas informasi di ImageNet juga berguna untuk mengidentifikasi kelas baru. Berikut gambaran arsitektur *INCEPTIONV3* pada gambar 2.6.

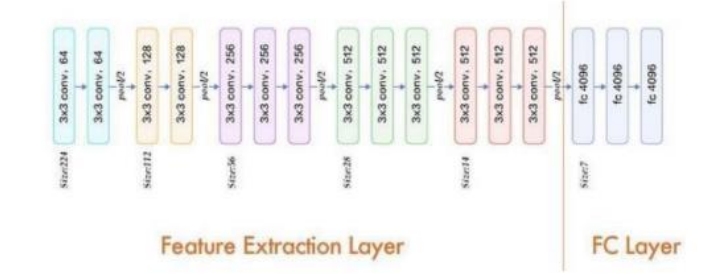


Gambar 2. 5 Arsitektur *InceptionV3*

Gambar 2.6 merupakan arsitektur *INCEPTIONV3*. Model ini terdiri dari: lebih dari 20 juta parameter, dan telah dilatih oleh salah satu perangkat keras terbaik di industri ahli. Model itu sendiri terdiri dari blok bangunan simetris dan asimetris, di mana: setiap blok terdiri dari berbagai *convolutional*, *average*, dan *max pooling*, *concats*, *dropouts*, dan lapisan yang terhubung penuh. Selain itu, normalisasi batch biasanya digunakan dan diterapkan pada input lapisan aktivasi ke dalam model ini. Klasifikasi dilakukan dengan menggunakan *Softmax*.

2.2.3.3 VGG16

VGG16Net merupakan model jaringan saraf konvolusi yang diusulkan oleh K. Simonyan dan A. Zisserman dari Universitas Oxford dalam tulisan "Very Deep *Convolutional* Networks for Large-Scale Image Recognition"[29]. Model VGG16 mencapai akurasi pengujian ke 5 yaitu 92,7% di ImageNet yang merupakan dataset lebih dari 14 juta gambar dengan 1000 kelas. Salah satu model yang terkenal diajukan ke ILSVRC-2014.



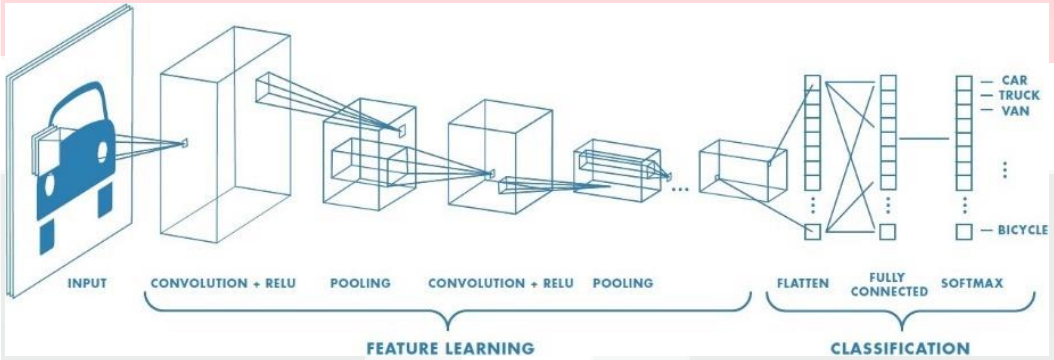
Gambar 2. 6 Arsitektur VGG16

Pada VGG16 memiliki 16 lapisan seperti namanya, dengan 4096- 4096-4096 neuron pada hidden layer dan 1000 neuron pada output layer. Pada tiga lapisan Fully-Connected (FC) mengikuti tumpukan lapisan konvolusional (yang memiliki kedalaman berbeda dalam arsitektur yang berbeda) yaitu dua yang pertama memiliki ukuran masing-masing 4096 neuron kanal, yang ketiga melakukan klasifikasi ILSVRC 1000 keluaran dan dengan demikian memuat 1000 saluran (satu untuk setiap kelas). Lapisan terakhir adalah lapisan soft-max. Konfigurasi fully connected layer adalah sama di semua jaringan. Sedangkan pada hidden layer dilengkapi dengan fungsi aktivasi ReLU.

2.2.4 Deep Learning & Convolutional neural network (CNN)

Deep Learning adalah bagian dari metode *machine Learning* yang menggunakan *Deep Neural Network* berfungsi untuk mengatasi masalah yang ada pada *machine Learning* [30]. Salah satu neural network yang dapat digunakan untuk mengenali atau mendeteksi sebuah objek pada gambar adalah *Convutional Neural Network (CNN).*

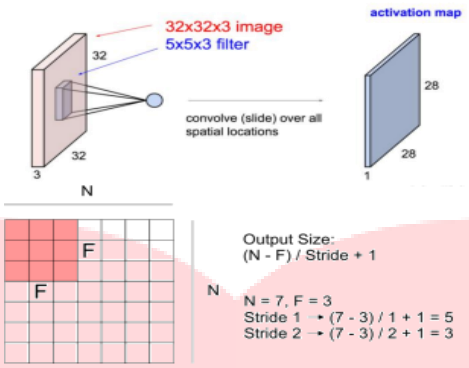
Convolutional Neural Network merupakan salah satu jenis neural network yang biasanya digunakan dalam pengolahan data *image*. Konvolusi atau biasa yang disebut dengan *convolution* adalah matriks yang memiliki fungsi melakukan filter pada gambar[31]. *Convolutional Neural Network* mempunyai beberapa layer yang difungsikan untuk melakukan filter di setiap prosesnya. Prosesnya dianggap menggunakan proses pelatihan. pada proses pelatihan terdapat tiga tahapan yaitu *Convolutional Layer, Pooling Layer*, serta *Fully Connected Layer*.



Gambar 2. 7 Arsitektur *Convolutional neural network*

2.2.4.1 *Convolutional Layer*

Seluruh data yang menyentuh lapisan konvolusional akan mengalami proses konvolusi. lapisan akan mengkonversi setiap filter ke seluruh bagian data masukan dan menghasilkan sebuah activation map atau feature map 2D. Filter yang terdapat pada *Convolutional Layer* memiliki panjang, tinggi(pixels) dan tebal sesuai dengan channel data masukan. Setiap filter akan mengalami pergeseran dan operasi “dot” antara data masukan dan nilai dari filter. Lapisan *Convolutional* secara signifikan mengalami kompleksitas model melalui optimalisasi outputnya. Hal ini dioptimalkan melalui tiga parameter, depth, *stride* dan pengaturan *zero padding*[32].



Gambar 2. 8 *Convolutional Layer* [38]

2.2.4.2 Pooling Layer

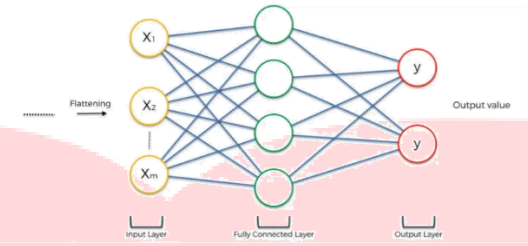
Pooling Layer merupakan tahap setelah *Convolutional Layer*. *Pooling Layer* terdiri dari sebuah filter dengan ukuran dan stride tertentu. Setiap pergeseran akan ditentukan oleh jumlah stride yang akan digeser pada seluruh area *feature map* atau activation map. Dalam penerapannya, *Pooling Layer* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Sebagai contoh, apabila kita menggunakan *Max Pooling* 2x2 dengan *Stride* 2, maka pada setiap pergeseran filter, nilai yang diambil adalah nilai yang terbesar pada area 2x2 tersebut, Sedangkan *Average Pooling* akan mengambil nilai rata-rata[33].

Gambar 2. 9 Pooling Layer [33]

2.2.4.3 Fully Connected

Feature map yang dihasilkan oleh tahap sebelumnya berbentuk multidimensional array. Sehingga, Sebelum masuk pada tahap *Fully Connected Layer*, *Feature Map* tersebut akan melalui proses " *flatten* " atau *reshape*. Proses flatten menghasilkan sebuah vektor yang akan digunakan sebagai input dari *Fully Connected Layer*. *Fully Connected Layer* memiliki

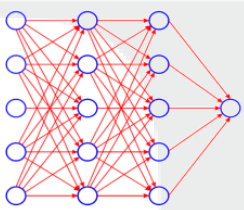
beberapa *Hidden Layer, Action Function, Output Layer dan Loss Function*[33].



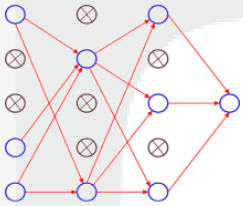
Gambar 2. 10 *Fully Connected Layer* [33]

2.2.4.4 Dropout

Dropout merupakan salah satu usaha untuk mencegah terjadinya *Overfitting* dan juga mempercepat proses *Learning* [34]. *Overfitting* adalah kondisi hampir semua data yang telah melalui proses *training* mencapai persentase yang baik, tetapi terjadi ketidaksesuaian pada proses prediksi. Dalam sistem kerjanya, *Dropout* menghilangkan sementara suatu neuron yang berupa *Hidden* Layer maupun *Visible Layer* yang berada di dalam jaringan[35].



Gambar 2. 12 Sebelum *Dropout* [35]

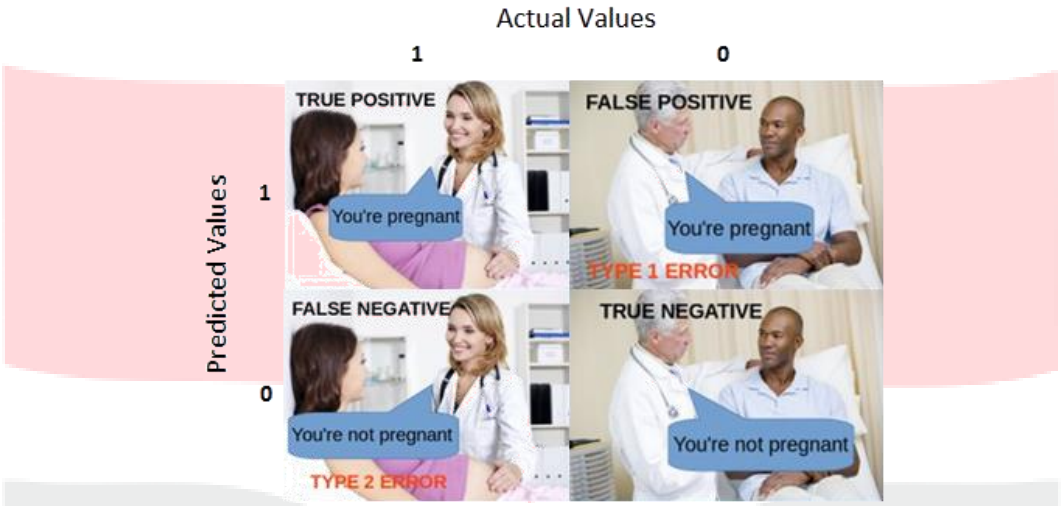


Gambar 2. 11 Setelah *Dropout* [35]

2.2.5 Confusion Matrix

Confusion Matrix sebuah ringkasan hasil dari pada masalah klasifikasi algortima. *Counfusin matrix* menunjukan cara model klasifikasi algortima bingung saat membuat prediksi. *Confusion Matrix* adalah ukuran kinerja untuk masalah klasifikasi machine *Learning* dimana outputnya bisa dua kelas atau lebih. *Confusion Matrix* adalah tabel dengan 4 kombinasi nilai prediksi dan nilai aktual yang berbeda. Ada empat istilah yang merupakan representasi hasil proses klasifikasi pada *Confusion Matrix* yaitu True *Positive*(TP), True *Negative*(TN), *False Positive*(FP), dan *False Negative*(FN)[36]. Berikut adalah contoh dalam memahami TP, TN, FP,

dan FN yang di analogikan sebagai hal analogi kehamilan pada gambar 2.14.



Gambar 2. 13 Analogi confusion matrix [39]

Pada gambar diatas untuk *true Positive* menjelaskan interpretasinya memprediksi bahwa seorang wanita hamil dan dia benar – benar hamil. *true negative* menjelaskan interpretasinya memprediksi bahwa pria tidak hamil dan sebenarnya dia tidak hamil. *False positive* merupakan sebuah kesalahan dari *true negative* dimana menjelaskan interpretasinya bahwa seorang pria hamil tetapi sebenarnya tidak. *False negative* merupakan kesalahan dari *true negative* yang menjelaskan interpretasinya bahwa seorang wanita tidak hamil tapi dia benar – benar hamil.

Dalam kinerja *Confusion Matrix* untuk mengukur hasil dari algoritma yang telah dipakai. Pada pengukuran kinerja *Confusion Matrix* ada beberapa yaitu *Accuracy, MissClassification, Precision, Recall* dan *F1-score atau F-Measure*[37]. Berikut penjelasan mengenai rumus untuk mengukur kinerja algoritma :

1. *Accuracy*

   Dalam perhitungan akurasi mengklasifikasikan seberapa akurat prediksi benar dengan keseluruhan data. Berikut rumusnya sama dengan 2.1 :

$$Accuracy \frac{TP+TN}{TP+TN+FP+FN} \quad (2.1)$$

2. *Precision*

   Dalam perhitungan *Precision* mengukur keakuratan antara data yang diinginkan dengan hasil yang telah diprediksi dari algoritma yang telah dipakai. Berdasarkan pernyataan tersebut *Precision* adalah sebuah rasio prediksi yang diperoleh dengan persamaan dari keseluruhan hasil nilai yang di prediksi *Positive*. Berikut rumusnya sama dengan 2.2 :

$$Precision \frac{TP}{TP+FP} \quad (2.2)$$

3. *Recall*

   Dalam perhitungan *Recall* mengukur keberhasilan algoritma dalam menemukan kembali sebuah informasi. Berdasarkan pernyataan tersebut maka *Recall* adalah sebuah rasi yang data keseluruhannya data yang benar *Positive*. Berikut rumusnya sama dengan 2.3:

$$Recall \frac{TP}{TP+FN} \quad (2.3)$$

4. F1-Score

   Nilai F1-Score atau dikenal juga dengan nama F-Measure didapatkan dari hasil Precision dan Recall antara kategori hasil prediksi dengan kategori sebenarnya.

$$F1 = \frac{2\ x\ Recall\ x\ precission}{Recall+Precission} \quad (2.4)$$