BAB 3. LANDASAN TEORI

3.1 Pengenalan Wajah

Pengenalan wajah merupakan proses untuk mengenali wajah seseorang baik dilakukan oleh manusia maupun dengan bantuan aplikasi komputer. Pengenalan wajah yang dilakukan dengan bantuan aplikasi komputer digunakan sebagai sistem keamanan, pengawasan, dan interaksi cerdas antara manusia dengan komputer [26]. Pada umumnya sistem pengenalan wajah terbagi atas dua jenis, yaitu sistem *feature based* dan *image based* [27].

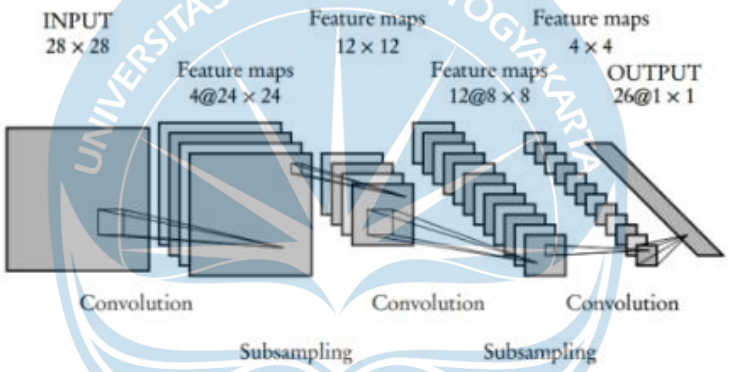
Sistem pengenalan dengan *feature based* merupakan sistem yang melakukan ekstraksi *feature* terlebih dahulu seperti memetakan komponen mata, hidung, mulut, dan komponen lainnya sebelum dilakukan pengenalan wajah [27]. Fitur-fitur pada area wajah manusia berbeda-beda, sehingga komputer dapat membedakan antara wajah yang satu dengan wajah yang lainnya dengan fitur-fitur tersebut [26]. Selain itu juga, pengenalan wajah dapat dilakukan dengan *image based* sistem. Sistem ini mengenali wajah menggunakan informasi mentah dari *piksel* citra gambar wajah. Salah satu metode yang menggunakan *image based* adalah PCA [27].

3.2 Preprocessing Gambar

Preprocessing adalah salah satu teknik untuk meningkatkan akurasi pada saat melakukan klasifikasi gambar menggunakan CNN [28]. Tujuan utama dari dilakukannya *preprocessing* adalah untuk menghilangkan data-data yang tidak dibutuhkan atau *noise* sehingga CNN lebih mudah dalam mengenali *object* yang dimaksud. Data noise biasanya berupa *background* dari *object* yang akan dikenali. *Preprocessing* dapat dilakukan dengan cara manual maupun dengan *programming*. Langkah-langkah dalam melakukan *preprocessing* bergantung pada tujuan dari model CNN tersebut dibuat. Pada penelitian ini, penulis melakukan deteksi wajah terlebih dahulu, kemudian memotong foto berdasarkan hasil deteksi tersebut sehingga gambar yang dimasukkan ke dalam model hanya memuat wajah dan memiliki sedikit *noise* di dalamnya.

3.3 Convolutional *Neural Network* (CNN)

Convolutional Neural Network atau biasa disebut CNN merupakan variasi dari *Multilayer Perceptron* atau jaringan syaraf tiruan manusia dengan banyak *layer*. Salah satu contoh arsitektur dari CNN dapat dilihat pada Gambar 3.1. Penelitian awal yang menggunakan metode ini adalah penelitian mengenai visual *cortex* pada indra penglihatan kucing yang diteliti oleh Hubel dan Wiesel [11]. Pada saat ini, metode CNN biasanya digunakan untuk melakukan pengelolaan data gambar. Perbedaan antara CNN dengan metode lainnya adalah konvolusi atau biasa disebut dengan *convolution*. Konvolusi merupakan matriks yang berfungsi melakukan filter pada gambar [29].



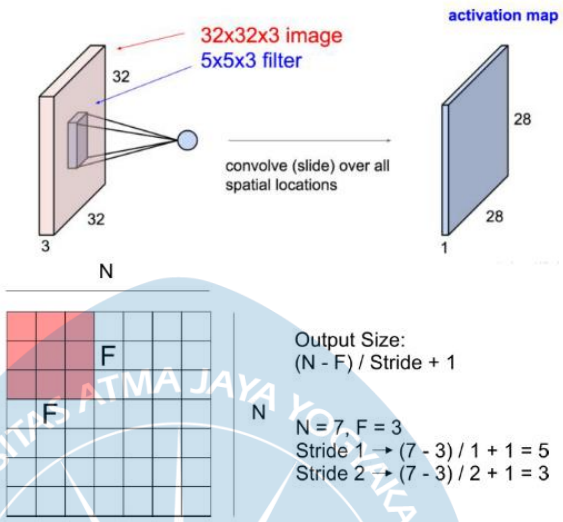
Gambar 3.1 Arsitektur CNN [30]

Dalam melakukan proses *filtering*, CNN memiliki dua matriks yaitu matriks pada *value input* dan matriks pada *kernel*. Pada proses *training*, CNN memiliki beberapa *layer* yang digunakan untuk melakukan fungsi *filtering* yaitu *convolution layer*, *pooling layer*, dan *fully connected layer* [31]. Adapun filter yang dimiliki oleh CNN dalam melakukan *filtering* yaitu :

1. *Convolution Layer*

   Convolution layer melakukan operasi konvolusi pada nilai keluaran yang dihasilkan oleh *layer* sebelumnya. Pada tahap ini, seluruh data yang dimasukkan akan menghasilkan sebuah *activation map* atau *feature map* 2D. *Activation map* atau *feature map* merupakan hasil dari filter yang dilakukan oleh *convolution layer*. Lapisan konvolusi memiliki tiga parameter yang terdiri dari *depth*, *stride*, dan pengaturan

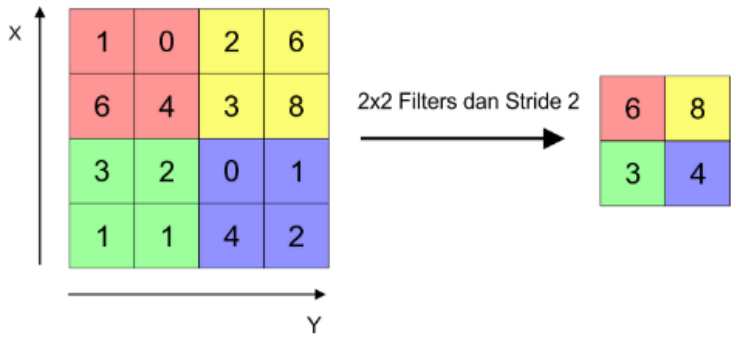
zero padding untuk melakukan optimalisasi terhadap model yang dibuat [29]. Proses yang terjadi dalam *convolution layer* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Convolution Layer [29]

2. *Pooling Layer*

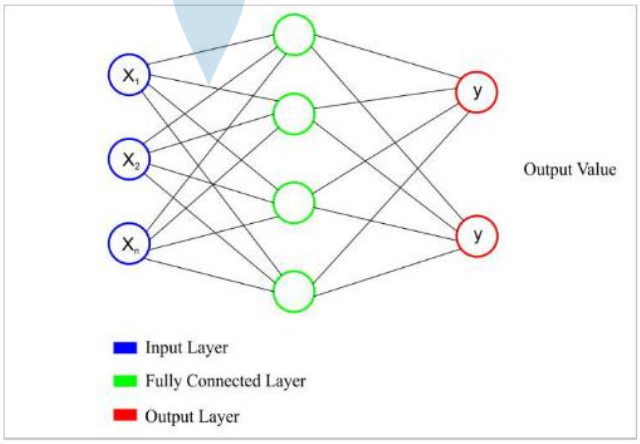
Pooling layer merupakan *layer* selanjutnya dari *convolution layer*. *Layer* tersebut memiliki filter dengan ukuran dan *stride* tertentu. Filter tersebut berfungsi untuk membuat matriks berdasarkan metode *pooling* yang digunakan. Metode *pooling* yang biasa digunakan adalah *max pooling* dan *average pooling*. Kedua metode tersebut akan mengambil nilai dari ukuran *pooling layer* yang telah ditentukan sebelumnya. Perbedaannya terletak pada cara metode tersebut mengambil nilai pada *activation map* yang dihasilkan oleh *convolution layer*. Apabila ukuran *pooling layer* adalah 2x2 dan *stride* adalah 2, maka dalam penerapan *max pooling*, setiap pergeseran filter akan diambil nilai terbesar pada area 2x2, sedangkan *average pooling* akan mengambil nilai rata-rata dari area tersebut [29]. *Pooling layer* dengan model *max pooling* dapat dilihat pada Gambar 3.3.



Gambar 3.3 Pooling Layer (Max Pooling) [29]

3. *Fully Connected Layer*

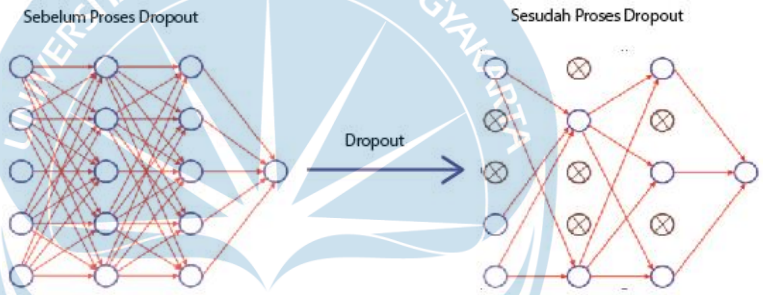
Feature map atau *activation map* yang dihasilkan oleh *layer* sebelumnya berbentuk multidimensional *array* sehingga harus dilakukan proses *flatten* atau *reshape* sebelum masuk ke tahap *fully connected layer*. Proses *flatten* akan menghasilkan sebuah vektor yang digunakan sebagai *input* pada *fully connected layer*. *Layer* tersebut memiliki beberapa sub *layer* yang terdiri atas *hidden layer*, *action function*, *output layer*, dan *loss function* [29]. *Layer* ini akan menghubungkan setiap jaringan *neuron* yang dihasilkan oleh *layer-layer* sebelumnya ke *layer* yang lainnya [31]. *Fully connected layer* dapat dilihat pada Gambar 3.4.



Gambar 3.4 Fully Connected Layer [29]

4. *Dropout*

Dengan keberadaan *layer* yang banyak, proses *learning* dapat menjadi lambat, oleh sebab itu perlu dilakukan proses *dropout*. Proses ini selain dapat mempercepat proses *learning* juga dapat digunakan untuk mencegah terjadinya *overfitting*. *Overfitting* merupakan kondisi saat akurasi yang diperoleh pada proses *training* memasuki persentase yang baik, tetapi terjadi kesalahan pada saat proses prediksi [29]. Pada penerapannya, *dropout* akan menghilangkan sementara neuron yang dipilih secara acak baik *hidden layer* maupun *visible layer*. Setiap *neuron* akan diberikan probabilitas yang bernilai 0 hingga 1.0 [31]. Proses *dropout* dapat dilihat pada Gambar 3.5.



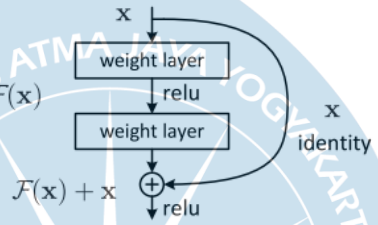
Gambar 3.5 Dropout [29]

3.4 Transfer Learning

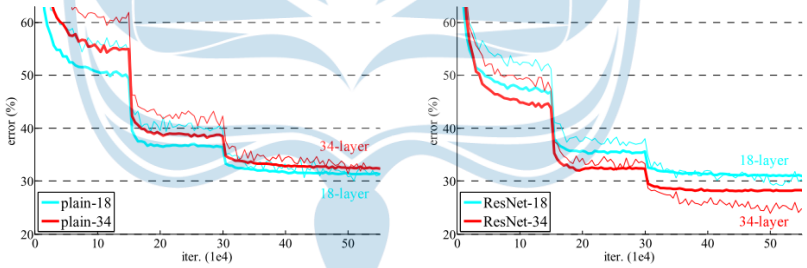
Transfer learning merupakan suatu teknik yang digunakan dalam *machine learning* untuk menggunakan *pretrain* model yang sudah pernah dibuat sebelumnya dan mampu menyelesaikan suatu masalah, kemudian digunakan kembali untuk menangani masalah yang serupa [32]. Pada pemanfaatannya, model yang sudah ada akan disesuaikan dengan permasalahan yang baru. Biasanya penyesuaian dilakukan dengan mengubah *output* dan juga penyesuaian terhadap *dataset* yang digunakan. Adapun beberapa arsitektur model yang mendukung *pretrain* model yaitu ResNet, SeNet, GoogleNet, dan VGGNet. Namun pada penelitian ini, penulis menggunakan arsitektur ResNet dengan varian ResNet 10 dan ResNet 50, arsitektur SeNet dengan varian Se-ResNet-50, dan VGG 16. Beberapa *pretrain* model yang penulis gunakan yaitu VGGFace, VGGFace2, dan ImageNet.

3.5 Residual Network atau ResNet

Residual network merupakan salah satu model arsitektur *deep convolusional network*. Model ini dibangun dengan pemahaman bahwa semakin banyak *layer* yang disusun secara beruntun tidak menjamin peningkatan akurasi, sehingga dalam melakukan penambahan *layer*, ResNet menghubungkan *layer* baru dengan salinan *layer* sebelumnya seperti pada Gambar 3.6. Penerapan metode tersebut terbukti memberikan penurunan *error rate* dibandingkan dengan penyusunan *layer* secara beruntun atau disebut juga *plain network*. Penurunan tersebut dapat dilihat pada Gambar 3.7 [24].



Gambar 3.6 Blok Layer Residual Network [24]

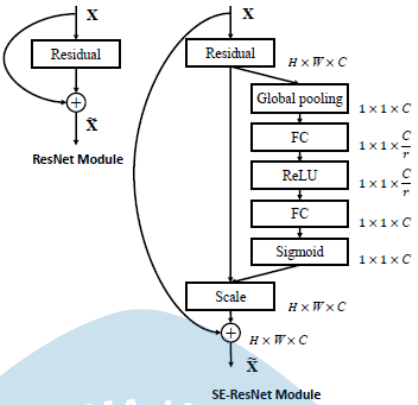


Gambar 3.7 Perbandingan Error Rate Antara Plain Network dan ResNet [24]

3.6 Squeeze-and-Excitation Networks atau SeNet

SeNet merupakah salah satu dari *deep convolutional network* seperti halnya ResNet. SeNet memiliki konsep pemetaan *layer* yang mirip dengan ResNet, yaitu membuat blok. Pada arsitektur SeNet, blok yang dibuat dinamakan Blok SE. Blok tersebut dirancang untuk memiliki suatu informasi global. Informasi tersebut dapat dipelajari oleh *layer-layer* selanjutnya agar dapat lebih selektif dalam memilih informasi yang berguna sekaligus mampu untuk fokus dalam mengelola informasi yang paling berguna. Dengan konsep blok tersebut, arsitektur ini dapat dikombinasikan dengan arsitektur yang lainnya, salah satu

arsitektur yang dapat dikombinasikan dengan arsitektur ini adalah ResNet, adapun hasil kombinasi *layer* pada ResNet dapat dilihat pada Gambar 3.8 [33].



Gambar 3.8 Arsitektur Blok ResNet (Kiri) dan Kombinasi Blok ResNet dengan Blok SE dari SeNet (Kanan) [33]

3.7 VGGFace 2

VGGFace merupakan *dataset* gambar wajah dalam jumlah yang sangat banyak. VGGFace sudah melalui dua tahap pengembangan, pada awalnya, gambar yang dikumpulkan berjumlah 2.6 juta dengan 2.622 identitas yang berbeda-beda. Kemudian pada tahun 2018, VGGFace memasuki tahap publikasi yang kedua atau biasa disebut dengan *VGGFace2*, memiliki 3.31 juta gambar dengan 9.131 identitas yang berbeda-beda. VGGface awalnya diuji dengan arsitektur VGG 16 atau biasa dikenal dengan nama *Deep Face* [18], sedangkan *VGGFace2* diuji menggunakan arsitektur ResNet 50 dan SeNet 50. Berdasarkan hasil pengujian *VGGFace2* terhadap *dataset IJB-A*, arsitektur SeNet 50 memiliki akurasi yang paling tinggi dibandingkan dengan ResNet 50, adapun hasil dari pengujian tersebut dapat dilihat pada Gambar 3.9 [25].

Training dataset	Arch.	1:1 Verification TAR			1:N Identification TPIR				
		FAR=0.001	FAR=0.01	FAR=0.1	FPIR=0.01	FPIR=0.1	Rank-1	Rank-5	Rank-10
VGGFace [17]	ResNet-50	0.620 ± 0.043	0.834 ± 0.021	0.954 ± 0.005	0.454 ± 0.058	0.748 ± 0.024	0.925 ± 0.008	0.972 ± 0.005	0.983 ± 0.003
MS1M [7]	ResNet-50	0.851 ± 0.030	0.939 ± 0.013	0.980 ± 0.003	0.807 ± 0.041	0.920 ± 0.012	0.961 ± 0.006	0.982 ± 0.004	0.990 ± 0.002
VGGFace2	ResNet-50	0.895 ± 0.019	0.950 ± 0.005	0.980 ± 0.003	0.844 ± 0.035	0.924 ± 0.006	0.976 ± 0.004	0.992 ± 0.002	0.995 ± 0.001
VGGFace2_ft	ResNet-50	0.908 ± 0.017	0.957 ± 0.007	0.986 ± 0.002	0.861 ± 0.027	0.936 ± 0.007	0.978 ± 0.005	0.992 ± 0.003	0.995 ± 0.001
VGGFace2	SENet	0.904 ± 0.020	0.958 ± 0.004	0.985 ± 0.002	0.847 ± 0.051	0.930 ± 0.007	0.981 ± 0.003	**0.994 ± 0.002**	**0.996 ± 0.001**
VGGFace2_ft	SENet	**0.921 ± 0.014**	**0.968 ± 0.006**	**0.990 ± 0.002**	**0.883 ± 0.038**	**0.946 ± 0.004**	**0.982 ± 0.004**	0.993 ± 0.002	0.994 ± 0.001
Crosswhite *et al.* [6]	-	0.836 ± 0.027	0.939 ± 0.013	0.979 ± 0.004	0.774 ± 0.049	0.882 ± 0.016	0.928 ± 0.010	0.977 ± 0.004	0.986 ± 0.003
Sohn *et al.* [20]	-	0.649 ± 0.022	0.864 ± 0.007	0.970 ± 0.001	-	-	0.895 ± 0.003	0.957 ± 0.002	0.968 ± 0.002
Bansalit et al. [4]	-	0.730†	0.874	0.960†	-	-	-	-	-
Yang *et al.* [25]	-	0.881 ± 0.011	0.941 ± 0.008	0.978 ± 0.003	0.817 ± 0.041	0.917 ± 0.009	0.958 ± 0.005	0.980 ± 0.005	0.986 ± 0.003

Gambar 3.9 Hasil Pengujian VGGFace2 dengan Dataset IJB-A [25]