BAB II

TINJAUAN PUSTAKA

2.1 Kacang Hijau

Kacang hijau (Vigna radiate L.) merupakan tanaman kacang-kacangan yang banyak ditanam oleh petani di Indonesia. Kacang hijau merupakan tanaman palawija yang memiliki banyak varietas dan merupakan salah satu komoditi yang penting karena menghasilkan bahan pangan (Leatemia *et al*, 2011 dalam Rahmat, 2018). Tanaman kacang hijau merupakan tanaman berumur pendek (60 hari) setelah masa tanam. Tanaman kacang hijau tumbuh dengan ukuran yang cukup pendek, memiliki akar tunggang, batang dengan cabang yang kecil dan tegak, berbulu, berwarna hijau dan kecoklatan, daun majemuk dengan tiga helai daun pertangkai, bunga yang berbentuk seperti kupu-kupu dan buah yang berbentuk polong. Biji atau buah dari tanaman kacang hijau ini berbentuk bulat dengan berat setiap butirnya biasanya sekitar 0,5 mg – 0,8 mg serta memiliki ukuran yang lebih kecil dibandingkan tanaman kacang yang lain seperti kedelai dan kacang tanah.

2.2 Manfaat Kacang Hijau

Tanaman kacang hijau memiliki banyak manfaat mulai dari pohonnya hingga bijinya. Pohon kacang hijau selain dapat digunakan sebagai pakan ternak juga bisa digunakan sebagai pupuk organik. Biji atau buahnya dapat digunaan sebagai bahan pangan (dikonsumsi sebagai bubur, sayur (toege) maupun isian kue).

Kacang hijau memiliki banyak zat yang bermanfaat untuk manusia selain mengandung banyak gizi, vitamin dan protein menurut (Purwono dan Hartono, 2005 dalam Rahmat, 2018) manfaat lain dari tanaman kacang hijau selain digunakan dalam pengobatan hepatitis, terkilir, beri-beri, demam nifas, kepala pusing (vertigo) juga digunakan sebagi pemulihan kesehatan seperti kencing kurang lancer, kurang darah, dan penyakit jantung.

Kacang hijau memiliki beberapa kelebihan jika dibandingkan dengan tanaman kacang-kacangan lain. Dalam (Juwita, 2014) menyebutkan kelebihan tersebut yaitu:

1. Lebih tahan kekeringan,
2. Hama dan penyakit yang menyerang relatif sedikit,
3. Dapat dipanen pada waktu yang relatif cepat, yaitu umur 55- 60 hari,
4. Cara tanam dan pegelolaannya di lapangan serta perlakuan pascapanennya relatif mudah,
5. Risiko kegagalan panen secara total relatif kecil,
6. Dapat dikonsumsi langsung dengan cara pengolahan yang mudah.

2.3 Kualitas

Salah satu faktor terpenting yang dapat meningkatkan produktivitas hasil panen dalam pertanian adalah benih. Selain tanah yang subur, pupuk, air, cahaya dan iklim, pemilihan benih yang bermutu atau berkualitas baik menjadi salah satu syarat mutlak dalam meningkatkan hasil panen. Benih yang berkualitas baik akan menghasilkan tanaman yang berkualitas, sehingga mampu menghasilkan panen

dengan produktivitas tinggi dan juga berkualitas. Secara umum, komponen kualitas benih dibedakan menjadi 4 (BBPPMBTPH, 2020) yaitu:

1. Komponen dengan mutu fisik, merupakan kondisi dari suatu benih yang berkaitan dengan warna, bentuk, ukuran, bobot, tekstur permukaan, tingkat kerusakan fisik, kebersihan dan keberagaman.
2. Komponen dengan mutu fisiologis, merupakan kondisi dari suatu benih yang berkaitan dengan daya hidup pada benih jika ditumbuhkan (dikecambahkan), baik dengan kondisi yang menguntungkan maupun kurang menguntungkan.
3. Komponen dengan mutu genetic, yaitu kondisi dari suatu benih yang berkaitan dengan kebenaran dari varietas benih, baik secara fenotip (fisik) ataupun genetiknya.
4. Komponen dengan mutu pathologis, yaitu berkaitan dengan ada atau tidak serangan penyakit pada suatu penyakit pada benih dan tingkat serangan yang terjadi.

2.4 Varietas Kacang Hijau

Kacang hijau memiliki berbagai varietas salah satunya adalah varietas vima. Varietas vima merupakan varietas kacang hijau yang banyak dibudidayakan petani dan memiliki daya adaptasi yang cukup baik pada lahan/sawah (Hakim dan Suyamto, 2012). Terdapat beberapa jenis varietas vima salah satu varietas vima yang berkualitas ekspor dan banyak diminati untuk dibudidayakan oleh petani di Indonesia adalah varietas vima 5.

Benih kacang hijau vima 5 dapat menghasilkan hingga 2,3 ton/ha. Keunggulan dari vima 5 adalah bijinya berwarna hujau kusam dengan polong berwarna coklat, memiliki daya adaptasi yang luas terhadap beberapa mikro iklim, polongnya lebih panjang, lebih tahan terhadap penyakit embun tepung dan hama thrips yang dapat menyebabkan kegagalan panen, bijinya sangat bagus dalam pembuatan bahan makanan dan lain sebagainya (tokofora.com, 2020).

2.5 Klasifikasi

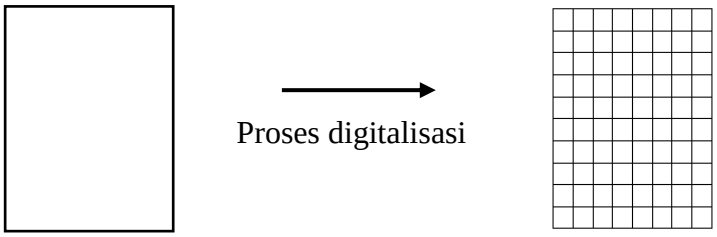
Klasifikasi merupakan suatu proses yang dilakukan untuk memperoleh suatu model atau fungsi yang menggambarkan dan membedakan konsep ataupun kelas data dengan tujuan untuk memprediksi kelas dari data yang belum dikenali kelasnya (J. Han M. Kamber, 2006 dalam Salsabila, 2018). Klasifikasi dapat digunakan dalam pengambilan suatu keputusan pada prediksi suatu kasus berdasarkan klasifikasi yang telah diperoleh. Algoritma yang digunakan untuk menyelesaikan masalah klasifikasi dapat dikategorisasikan ke dalam *supervised learning* atau pembelajaran yang diawasi yaitu data label atau target ikut berperan sebagai '*supervisor*' atau 'guru' yang mengawasi proses pembelajaran dalam mencapai tingkat akurasi atau presisi tertentu (Wibowo, 2017).

2.6 Citra (*Image*)

Citra (*image*) merupakan suatu gambar pada bidang dua dimensi yang tersusun dengan banyak piksel dari bagian terkecil citra. Pada umumnya, citra gambar terbentuk dari beberapa kotak persegi empat yang teratur sehingga jarak

horizontal dan *vertical* antar piksel sama pada seluruh bagian citra (ldya, *et al*, 2010 dalam Dewi, 2018). Citra dibagi menjadi dua yaitu citra yang bersifat analog dan digital. Citra yang bersifat analog tidak dapat dipresentasikan di dalam komputer, sehingga citra analog harus dikonversi terlebih dahulu menjadi citra digital agar dapat diproses dalam komputer (Sutojo, 2017 dalam Kusumaningrum, 2018).

Pengolahan citra dengan menggunakan komputer dapat dilakukan dengan mempresentasikan suatu citra (*kontinu*) secara numerik dengan nilai diskrit yang dapat disebut dengan digitalisasi atau citra digital. Citra merupakan suatu fungsi dua dimensi *f(x,y),* dimana *x* dan *y* adalah kooordinat spasial, dan *f(x,y)* adalah nilai pada koordinat *(x,y)* yang sering disebut intensitas. (Gonzalez dan Woods, 2008 dalam Dewi, 2018). Pada umumnya citra digital berbentuk empat persegi panjang dengan ukuran dimensinya dinyatakan dengan tinggi x lebar atau lebar x panjang. Berikut ini merupakan bentuk citra digital dari kotak persegi empat :



Gambar 2.1 Proses Digitalisasi Citra

Menurut (Dewi,2018) citra digital dapat dikategorikan dalam beberapa jenis, yaitu citra biner (*binary image*), citra keabuan (*greyscale)* dan citra warna. Citra berwarna tersusun atas tiga warna primer yaitu *red*, *green* dan *blue* atau disingkat (RGB). Dari setiap titik pada gambar atau citra

dengan bentuk lebar x tinggi (lebar x panjang) dalam ukuran *pixel* dapat mewakili kombinasi dari tiga komponen warna tersebut.

Citra digital mempunyai berbagai ukuran yang disebut *pixel*. Citra atau gambar biasanya juga direpresentasikan dengan matriks yang berukuran N baris dan M kolom. Elemen-elemen matriks pada citra digital yang berukuran N x M maka mempunyai NM buah pixel. Representasi dari citra atau gambar dalam bentuk matriks dapat dilihat pada persamaan berikut (Yusuf, 2017 dalam Putri, 2020):

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,1) & f(2,2) & \dots & f(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ f(M,1) & f(M,2) & \dots & f(M,N) \end{bmatrix} \quad (1)$$

Keterangan:

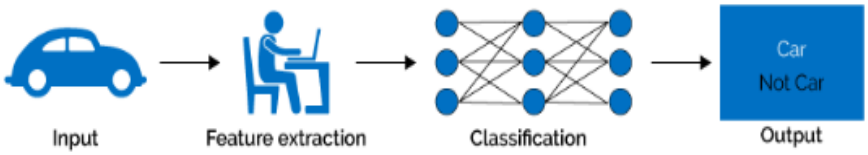
M = Banyaknya kolom pada *array* citra

N = Banyaknya baris pada *array* citra

2.6.1 *Machine Learning*

Mechine Learning (pembelajaran mesin) merupakan suatu teknologi atau algoritma yang dapat digunakan dalam berbagai bidang ilmu komputer. *Mechine Learning* juga merupakan pendekatan dalam *Artificial Intelligence* yang banyak digunakan untuk menirukan atau menggantikan perilaku manusia dalam menyelesaikan suatu masalah atau otomatisasi. Dalam penerapannya, pengguna dapat membuat sebuah model menggunakan bantuan *mechine learning* yang secara otomatis dapat mengenai pola, mengklasifikasi data dan membuat prediksi berdasarkan data yang dimasukkan (Naveen Sundar Govindarajulu, 2018 dalam

Farhani, 2020). *Mechine Learning* sebagai komparasi berdasarkan pengalaman dapat digunakan untuk membuat prediksi yang akurat atau juga dapat digunakan untuk meningkatkan performa (Izah, 2018). Berikut ini merupakan konsep dalam *mechine learning*:



Gambar 2.2 *Machine Learning*

Dalam pengaplikasian teknik-teknik dalam *mechine learnig* terdapat karakteristik yang harus dipenuhi yaitu adanya proses pelatihan, pembelajaran atau *training* dan adanya data. Pada data tersebut akan dibagi menjadi dua yaitu data t*raining* dan data *testing*, yang mana data *training* tersebut akan digunakan untuk melatih algoritma, sedangkan data *testing* digunakan untuk mengetahui performa algoritma yang telah dilatih sebelumnya ketika menemukan data baru yang belum pernah terlihat (Kusumaningrum,2018)

2.6.2 *Deep Learning*

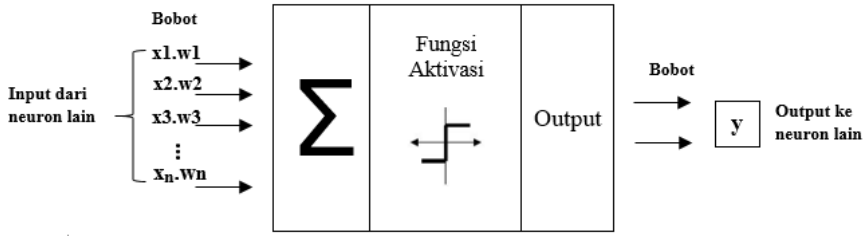
Deep Learning merupakan salah satu dari bagian *mechine learnig* yang memanfaatkan Jaringan Syaraf Tiruan (JST) untuk mengajarkan komputer dalam melakukan suatu tindakan dan mengimplementasikan permasalahan pada dataset yang besar. Menurut (Deng dan Yu, 2014 dalam Dewi, 2018). *Deep Learning* didefinisikan sebagai salah satu bidang *mechine learnig* dengan memanfaatkan banyak *layer* pengolahan informasi nonlinier dalam melakukan ekstraksi fitur, pengenalan pola serta klasifikasi. Teknik dalam *Deep Learning* menjadikan

arsitektur menjadi sangat kuat untuk *Supervised Learning* (Data pembelajaran mencakup keluaran yang sudah ditentukan). Penambahan lebih banyak lapisan menjadikan model pembelajaran yang bisa mewakili citra berlabel dengan lebih baik. (Shafira, 2018).

Konsep JST yang memiliki banyak lapisan yang dapat ditangguhkan pada algoritma *mechine learning* yang sudah ada sehingga computer dapat belajar dengan skala yang besar, kecepatan dan akurasi. Salah satu fitur *deep learning* yang sering digunakan dalam klasifikasi adalah *Feature Engineering* yaitu suatu teknik yang penting untuk mencapai hasil yang baik pada tugas prediksi dengan cara mengekstrak pola yang berguna dari data, sehingga bisa lebih mudah dalam membedakan kelas tersebut (Nurhikmat, 2018).

2.6.3 Komponen *Neural Network*

Hampir semua komponen yang dimiliki oleh jaringan syaraf sama, tetapi juga terdapat jaringan syaraf yag memiliki tipe yang berbeda. *Neural network* atau jaringan syaraf terdiri dari beberapa kumpulan neuron unit yang saling terhubung antar satu dengan yang lain. Masing-masing neuron mentransformasikan informasi yang telah diterima menuju neuron lain melalui sambungan atau *link.* Hubungan ini dikenal dengan sebutan bobot. Informasi tersebut disimpan pada suatu nilai tertentu dan pada bobot tertentu. Berikut ini merupakan struktur pada *neural network*:



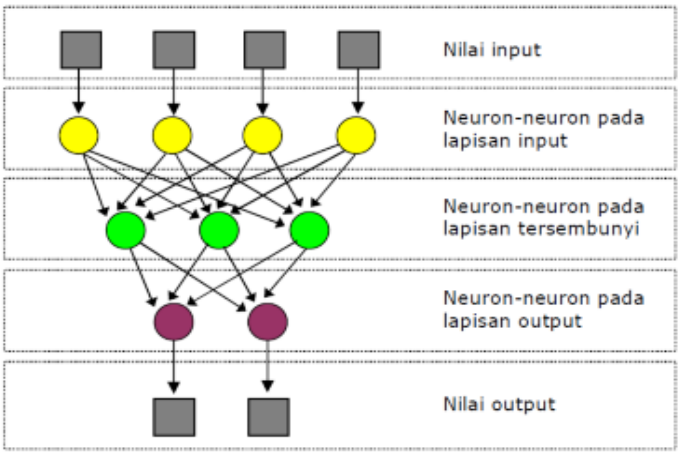
Gambar 2.3 Struktur *Neural Network*

Komponen yang dimiliki oleh *neural network* berdasarkan gambar 2.5 diatas memiliki struktur sebagai berikut (Nurhikmat, 2018):

1) *Input* dari variabel independent (X1, X2, X3, …, Xn) yang merupakan sebuah sinyal yang masuk ke sel syaraf.
2) Dalam struktur tersebut memliliki bobot yang terdiri dari beberapa bobot (W1, W2, W3,…,Wn) yang berhubungan dengan masing-masing node.
3) *Treshold* merupakan nilai ambang batas internal dari node. Besar nilai ini mempengaruhi aktivasi dari *ouput* node y.
4) *Activation Function* (Fungsi Aktivasi) merupakan operasi matematika yang dikenal pada sinyal output y.

Menurut (Shafira, 2018) Penempatan neuron-neuron pada jaringan syaraf akan dikumpulkan dalam *neuron layer* (lapisan-lapisan). Kemudian neuron pada satu lapisan akan di hubungkan dengan lapisan sebelum dan lapisan sesudahnya, kecuali pada lapisan input dan output. Informasi (*input*) yang masuk pada jaringan syaraf akan dirambatkan dari lapisan input sampai dengan lapisan *output* yang biasa disebut dengan lapisan tersembunyi (*hidden layer).* Pada beberapa jaringan syaraf ada yang tidak mempunyai lapisan lapisan tersembunyi dan ada juga yang disusun secara matriks neuronnya. Informasi bisa juga dirambatkan pada jaringan

secara mundur, tergantung pada algoritma pembelajaran yang digunakan. Berikut ini merupakan contoh gambaran jaringan syaraf dengan menggunakan tiga lapisan:



Gambar 2.4 Contoh Jaringan Syaraf dengan Tiga Lapisan

2.6.4 Fungsi Aktivasi

Fungsi aktivasi merupakan sebuah fungsi yang digunakan untuk menentukan aktif atau tidaknya neuron. Fungsi aktivasi membantu memecahkan model yang bersifat non-linier yang kompleks. Sehingga tanpa adanya fungsi aktivasi, jaringan *output* hanya akan menjadi fungsi *linear* dan jaringan syaraf tidak dapat mempelajari data kompleks seperti audio, gambar, dan lain-lain.

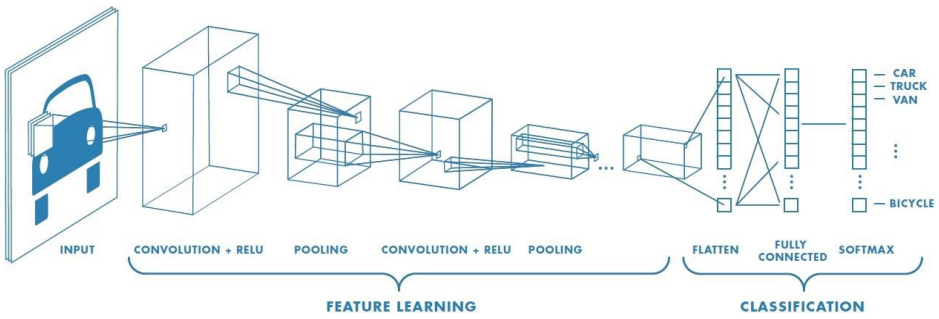
Beberapa fungsi aktivasi yang umum digunakan dalam jaringan syaraf yaitu berupa fungsi *linear* seperti ReLu (*Rektifikasi unit linier*) , atau fungsi non-linear seperti fungsi sigmoid atau logistik dan fungsi tanh (Garis singgung hiperbolik). Fungsi aktivasi yang banyak digunakan dalam model CNN adalah ReLu (*Rectified Linear Unit)* yang mengaplikasikan fungsi f(x)=max(0,x) dengan melakukan *thresholding* dari nilai nol terhadap nilai piksel pada input citra. Pada

aktivasi ini membuat seluruh nilai piksel yang bernilai kurang dari nol pada suatu citra akan dijadikan 0 (Shafira, 2018).

2.7 *Convolutional Neural Network*

Convolutional Neural Network (CNN*)* merupakan salah satu algoritma dari *deep learning* dan merupakan pengembangan dari *Multilayer Perceptron* (MLP) yang dirancang dan dapat digunakan untuk mengolah data terstruktur atau *grid* dalam bentuk dua dimensi/ 2D, misalnya gambar atau suara (Ilahiyah dan Nilogiri, 2018). *Convolutional Neural Networks* merupakan penerapan dari *Artificial Neural Networks* (ANN) yang saat ini dikenal sebagai model terbaik untuk memecahkan masalah dalam pengenalan objek atau *image*. CNN merupakan operasi konvolusi yang terdiri dari bebebrapa tahap atau lapisan pemrosesan yang terinspirasi dari sistem syaraf biologis (Hu, *et al*, 2015)

Menurut (Wulandari, *et al*, 2020) CNN juga digunakan sebagai klasifikasi data yang berlabel dengan menggunakan metode *supervised learning* yaitu terdapat data yang dilatih dan terdapat variabel yang ditargetkan sehingga tujuan dari ini yaitu untuk mengelompokkan suatu data ke dalam data yang sudah ada. Dalam arsitektur CNN data yang akan dilatih terdiri dari beberapa tahapan seperti masukkan (*input*) dan keluarkan (*output*) dari setiap tahapan terdiri dari beberapa *array* atau *feature map*. Tahapan tersebut dapat dilihat pada jaringan arsitektur CNN berikut ini:



Gambar 2.5. Arsitektur *Convolutional Neural Network*

Pada struktur CNN terdiri dari input, ekstraksi fitur, proses klasifikasi dan *output*. Dalam setiap lapisan akan menerima *input* gambar secara langsung dan menghasilkan *output* berupa vektor untuk diolah pada lapisan berikutnya. Pada proses ekstraksi fitur, struktur CNN terdiri dari beberapa lapisan tersembunyi (*hidden layer*) yaitu berupa lapisan konvolusi, fungsi aktivasi (ReLU), dan *pooling*. Metode CNN bekerja secara hirarki, sehingga *output* pada lapisan konvolusi pertama akan digunakan sebagai input pada proses konvolusi berikutnya. Sedangkan pada proses klasifikasi terdiri dari *fully-connected* dan fungsi aktivasi (*softmax*) yang outpunya berupa hasil klasifikasi (katole et al, 2015 dalam Arrofiqoh dan Harintaka, 2018)

2.7.1 Konvolusi

Proses konvolusi merupakan operasi aljabar linier yang menghasilkan matriks dari *filter* pada citra yang akan di proses. Proses tersebut dinamakan dengan lapisan konvolusi *(convolutional layer)* yang merupakan lapisan utama dan yang terpenting dalam satu jaringan. *Filter* yang terdapat pada *Convolutional Layer* memiliki panjang, tinggi dan tebal sesuai dengan volume data masukan. Konvolusi juga diartikan sebagai proses yang dilakukan untuk memperoleh suatu

piksel yang di dasarkan pada nilai piksel itu sendiri dan tetangganya dengan melibatkan suatu matriks yang disebut sebagai kernel serta mempresentasikan pembobotan (Kusumanto *et al*, 2011 dalam Ilahiyah dan Nilogiri, 2018). Operasi dalam proses ini merupakan suatu fungsi *output* sebagai *feature map* dari masukkan (*input*) citra. Masukan dan keluaran pada operasi konvolusi ini dapat dilihat sebagai dua *argument* yang bernilai riil. Operasi konvolusi ini dapat dituliskan dengan persamaan sebagai berikut (Shafira, 2018):

$$S(t) = (x^* w)(t) \qquad (2)$$

Keterangan :

S(t) = Fungsi hasil operasi konvolusi

X = Input

W = bobot (kernel)

Pada fungsi *s(t)* memberikan *output* tunggal berupa *feature map*. Argumen pertama adalah *input* yang merupakan x dan *argument* kedua adalah w yaitu sebagai kernel atau *filter*. Apabila dilihat dari *input* yang merupakan citra dimensi, maka dapat dikatakan t merupakan piksel dan dapat menggantinya dengan i dan j. Maka untuk itu, operasi untuk konvolusi ke *input* dengan lebih dari satu dimensi (Nurhikmat, 2018) dapat ditulis sebagai berikut:

$$S(i,j) = (K^* I)(i,j) = \sum_{\infty} \sum_{\infty} I\,(i-m, j-n) K(m,n) \qquad (3)$$

$$S(i,j) = (K^* I)(i,j) = \sum_{\infty} \sum_{\infty} I\,(i+m, j-n) K(m,n) \qquad (4)$$

Keterangan :

i dan J = Piksel dari sebuah citra

K = Kernel atau Fitur

I = Input dan Kernel

Sebagai alternatif, operasi konvolsui juga dapat dilihat sebagai perkalian matriks antara citra masukan dan kernel dimana keluarannya dihitung dengan *dot product.* kemudian selain itu, penentuan volume *output* juga dapat ditentukan dari masing-masing lapisan yaitu dengan menggunakan *hyperparameters*. *Hyperparameter* yang digunakan pada persamaan di bawah ini digunakan untuk menghitung seberapa banyaknya neuron aktivasi yang dihasilkan dalam sekali *output.* Persamaan menurut (Nurhikmat, 2018) tersebut adalah sebagai berikut:

$$(W - F + 2P)/(S + 1) \quad (5)$$

Keterangan :

W = Ukuran volume gambar

F = Ukuran Filter/ kernel

P = Nilai *Padding* yang digunakan (0)

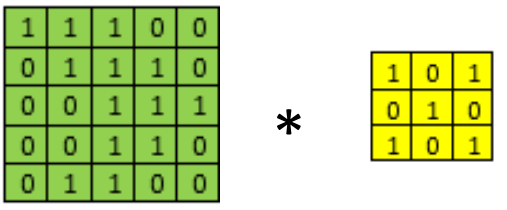
S = Ukuran Pergeseran (*Stride*)

Lapisan konvolusi secara signifikan mengalami kompleksitas model melalui optimalisasi outputnya. Hal ini dapat dioptimalkan melalui parameter yaitu:

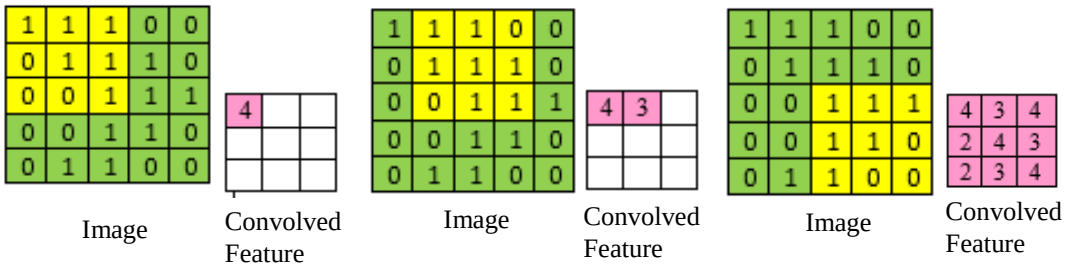
1. *Stride,* merupakan sebuah parameter yang digunakan dalam menentukan berapa jumlah pergeseran filter melalui *input* citra. Jika nilai *stride* adalah satu, maka filter akan bergeser sebanyak satu piksel secara *horizontal* lalu vertikal. Semakin kecil *stride* yang digunakan, maka akan semakin detail informasi yang diperoleh dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar.

2. *Zero Padding*, adalah suatu parameter yang digunakan dalam menentukan jumlah piksel (berisi nilai nol) yang akan ditambahkan di setiap sisi dari proses *input*. Hal ini digunakan untuk memanipulasi dimensi *output* dari *convolutional layer (feature map)*. Penggunaan *zero padding* ini juga memungkinkan untuk mengatur dimensi dari *output* agar tetap sama dengan dimensi *input*, atau setidaknya tidak berkurang secara drastis. Sehingga selanjutnya bisa dilakukan ekstraksi *feature* yang lebih mendalam. Selain itu, dengan penggunaan *zero padding* juga dapat menjadikan performa dari suatu model menjadi meningkat, karena *filter* akan fokus pada informasi yang sebenarnya yaitu berada di antara *zero padding* tersebut.

Proses konvolusi pada suatu gambar dengan ukuran 5x5 yang ditandai dengan warna hijau (citra yang akan dikonvolusi) serta ukuran kernel 3x3 yang ditandai dengan warna kuning dan *stride* 1 (kernel yang bergerak dari sudut kiri atas sampai kanan bawah) sehingga hasil dari proses konvolusi dapat dilihat pada gambar disebelah kanannya (*pink*). Ilustrasi proses konvolusi dapat dilihat pada gambar berilkut ini:



Gambar. 2.6 Ilustrasi *Image* 5x5 P*ixel* dan Kernel 3x3



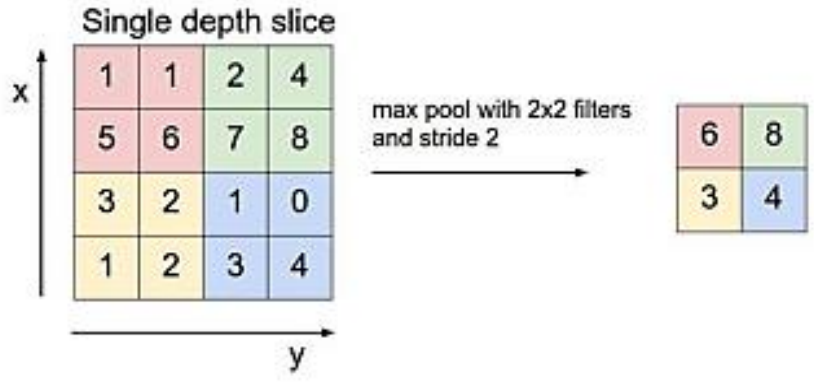
Gambar 2.7 Ilustrasi Proses Konvolusi

2.7.2 *Pooling Layer*

Pooling layer merupakan suatu lapisan fungsi dengan *feature map* sebagai *input* dan mengolahnya dengan berbagai macam operasi statistik dengan berdasarkan nilai piksel terdekat (Ilahiyah dan Nilogiri, 2018). *Pooling layer* juga dapat digunakan untuk mengambil nilai rata-rata (*average pooling)* dan nilai maksimal (*max-pooling)* yang diperoleh dari bagian piksel pada sebuah citra. Lapisan *pooling* pada arsitektur model CNN yang dimasukkan antara lapisan konvolusi secara berturut-turut dapat secara progresif mengurangi ukuran volume *output* pada *feature map,* sehingga jumlah parameter dan perhitungan pada jaringan akan berkurang, serta dapat digunakan untuk mengendalikan *overfitting* (Nurhikmat, 2018).

Lapisan *pooling* bekerja pada setiap tumpukan *feature map* dan dapat melakukan pengurangan pada ukurannya. Bentuk dari lapisan *pooling* yang paling umum adalah dengan menggunakan *filter* berukuran 2x2 yang kemudian diaplikasikan dengan langkah sebanyak dua dan kemudian dapat beroperasi disetiap irisan dari inputnya. Dengan bentuk seperti ini dapat mengurangi *feature map* sampai 75% dari ukuran aslinya. Berikut ini merupakan contoh gambar pada

operasi *max pooling* 2x2 dengan *stride* dua, maka di setiap pergeseran *filter* nilai *maximum* pada area 2x2 pixel tersebut yang akan terpilih, sedangkan *average pooling* akan memilih nilai rata-ratanya (Peryanto, *et al,* 2019).



Gambar 2.8 Proses *Max Pooling*

Dari proses *max pooling* pada gambar diatas lapisan *pooling* akan beroperasi disetiap lapisan irisan kedalam volume input secara bergantian. Pada gambar diatas proses *max pooling* dari ukuran input 4x4 pada masing-masing 4 angka pada input operasi tersebut diambil nilai maksimalnya dan dilanjutkan dengan membuat ukuran *output* baru menjadi ukuran 2x2. Pada kotak yang berwarna merah, hijau, kuning dan biru pada sisi kiri merupakan kelompok kotak yang akan dipilih nilai maksimalnya. Sehingga hasil dari proses tersebut dapat dilihat pada kotak di sebelah kanannya. Output dari proses *pooling* adalah matriks dengan dimensi yang berukuran lebih kecil dibandingkan dengan matrik awal. Proses konvolusi dan pooling dilakukan beberapa kali hingga didapat peta fitur dengan ukuran yang dikehendaki. Peta fitur yang dihasilkan tersebut akan menjadi *input* bagi *fully connected netural network.* dan akan dihasilkan *output class* dengan keluaran berupa akurasi kelas untuk klasifikasi. Tujuan dari p*ooling layer*

yaitu untuk mengurangi dimensi dari *feature map* (*downsampling*) sehingga dapat mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan dapat mengatasi *overfiting* (Peryanto, *et al* 2019).

2.7.3 *Fully-Conected Layer*

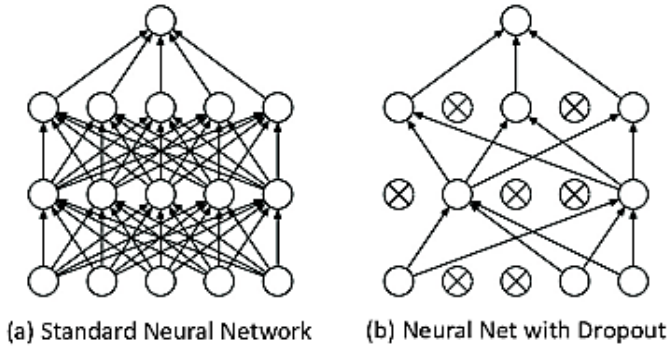
Fully-Conected Layer merupakan sebuah lapisan dimana semua neuron aktivasi pada lapisan sebelumnya terhubung dengan neuron di lapisan selanjutnya sama seperti halnya dengan jaringan syaraf / *neural network* biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dihubungkan pada semua neuron di *Fully-Conected Layer*. Pada umumnya *Fully-Conected Layer* digunakan dalam penerapan *Multi Layer Perceptron* (MLP) yang bertujuan untuk melakukan transformasi pada dimensi data agar dapat diklasifikasikan secara *linear* (Nurhikmat, 2018).

Perbedaan antara *fully connected layer* dan lapisan konvolusi biasa adalah neuron pada lapisan konvolusi dapat terhubung hanya ke daerah tertentu pada *input*, sedangkan pada *fully connected layer* mempunyai neuron yang secara keseluruhan akan terhubung. Akan tetapi, kedua lapisan tersebut masih mengoprasikan *produk dot*, sehingga memiliki fungsi tidak begitu berbeda (Shafira, 2018).

2.7.4 *Dropout Regularization*

Dropout di definisikan sebagai suatu teknik regularisasi dari jaringan syaraf tiruan dimana beberapa akan dipilih secara *random* dan tidak dipakai selama

pelatihan atau data latih. Neuron-neuron ini dapat dibuang juga secara *random.* Hal ini berarti bahwa kontribusi neuron yang dibuang akan dihentikan sementara jaringan dan bobot baru juga tidak ditetapkan pada neuron saat melakukan proses *backpropagation* (Peryanto, *et al* 2019). Berikut ini merupakan contoh gambaran pada proses *dropout*:



Gambar 2.9 *Dropout Regularization*

Gambar 2.9 merupakan contoh implementasi dari *dropout regularization* yaitu (a) merupakan jaringan syaraf biasa dengan 2 lapisan tersembunyi sedangkan (b) merupakan jaringan syaraf yang telah dilakukan regularisasi *dropout* dimana ada beberapa neuron aktivasi yang sudah tidak dipakai lagi. Teknik ini sangat mudah diaplikasikan pada model CNN dan akan memberikan dampak pada performa model dalam melatih serta mengurangi *overfitting* (Shafira, 2018).

2.7.5 *Softmax Classifier*

Softmax Classifier merupakan suatu bentuk lain atau regenerasi dari algoritma *Logistic Regression* yang dapat digunakan dalam pengklasifikasian yang lebih dari dua kelas. Standar klasifikasi yang biasa dilakukan oleh algoritma

Logistic Regression adalah klasifikasi kelas biner. Menurut (Ilahiyah, dan Nilogiri, 2018) *Softmax* mempunyai bentuk persamaan sebagai berikut:

$$f_j(z) = \frac{e^z j}{\sum_k e^z k} \quad (6)$$

Keterangan:

f_j = nilai output ke j

z = hipotesis yang diberikan oleh model pelatihan

$e^z j$ = nilai unit ke-j

$\sum_k$ = jumlah neuron lapisan output

$e^z k$ = nilai neuron ke-k

Selain itu *Softmax* juga memberikan hasil yang lebih intuitif dan memiliki interpretasi dalam probabilistik yang lebih baik dibandingkan dengan algoritma klasifikasi lainnya. *Softmax* dapat memungkinkan peniliti untuk dapat menghitung nilai probailitas untuk semua label. Dari hasil label yang ada, dapat diambil sebuah *vector* yang mempunyai nilai riil dan merubahnya menjadi *vector* dengan nilai antara nol dan satu, apabila semua dijumlahkan maka akan bernilai satu.

2.8 *Crossentropy Loss Function*

Loss Function atau *Cost Function* adalah suatu fungsi yang dapat menggambarkan kerugian terkait dengan semua kerugian yang dihasilkan oleh suatu model. *Loss Function* yang baik merupakan fungsi yang dapat menghasilkan *error* yang paling rendah. Disaat suatu model yang memiliki kelas yang cukup banyak, diperlukan adanya cara untuk dapat mengukur perbedaan antara probailitas dari hasil hipotesis dengan probabilitas kebenaran yang asli, dan

selama pelatihan terdapat banyak algoritma yang dapat menyesuaikan parameter sehingga perbedaan ini dapat diminimalkan. Dalam (Wulandari, *et al*,2020) dapat diketahui persamaan *loss function* dengan *crossentropy* adalah sebagai berikut:

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{r=1}^{M} Y_{r,n}\log(f(z_r)n) \qquad (7)$$

Dimana nilai 1 = kelas sesuai dan nilai 0 = kelas tidak sesuai

Keterangan :

N = Jumlah data citra

M = Jumlah kategori

f (Zr) = Fungsi *Softmax classifier*

Yr,n = Indikator biner

Gambaran umum dari algoritma ini yaitu meminimalkan kemungkinan *log negative* dari dataset, yang merupakan ukuran langsung dari performa prediksi dari model.

2.9 *Adaptive Moment Estimation (*Adam*)*

Adam adalah algoritma pengoptimalan yang dapat digunakan sebagai pengganti dari prosedur *stochastic gradient descent* (SGD) klasik untuk memperbarui bobot pada suatu jaringan berdasarkan pada training atau pelatihan. Adam juga merupakan algoritma yang popular dalam bidang *deep learning*, karena ia mencapai hasil yang baik dengan cepat. Persamaan dari algoritma optimasi adam adalah sebagai berikut (Anugerah, 2018):

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\,\hat{m}_t \qquad (8)$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad (9)$$

$$\widehat{v}_t = \frac{v_t}{1-\beta_1^t} \quad (10)$$

Keterangan:

η : merupakan *learning rate*

m_t : nilai decay rata-rata dari iterasi sebelumnya

v_t : nilai decay rata-rata yang dikuadratkan dari iterasi sebelumnya

β : nilai decay

$\in$: penambahan untuk menghindari pembagian dengan nol dimana $\epsilon = 1e - 8$

2.10 *Confusion Matrix*

Confusion Matrix merupakan suatu metode yang digunakan untuk mengukur performa dari suatu model klasifikasi dengan mengcari nilai *precision*, *recall*, dan nilai akurasi dari suatu model. Selain itu terdapat beberapa istilah umum yang dapat dipakai dalam proses pengukuran kinerja dari model klasifikasi yaitu:

1. *True Positive (TP*) : Data positif yang diprediksi benar
2. *True Negative (TN)* : Data *negative* yang diprediksi benar
3. *False Positive (FP)* : Data *negative* namun diprediksi sebagai data positif
4. *False Negative (FN)*: Data positif namun diprediksi sebagai data *negative*

Istilah-istilah tersebut dapat dirangkum sebagai suatu matriks yang disebut *confusion matrix* sebagaimana dapat ditunjukkan pada tabel berikut (Shafira, 2018):

Tabel. 2.1 *Confusion Matrix*

		Predicted class		
		Yes	No	Total
Actual class	Yes	TP	FN	P
	No	FP	TN	N
	Total	P	N	P+N

2.11 Akurasi, *Precision, Recall, dan F-Measure*

Akurasi merupakan persentase dari data uji yang diklasifikasikan dalam kelas yang benar. Akurasi digunakan sebagai parameter keakuratan suatu model dalam melakukan klasifikasi. Persamaan akurasi dapat dinyatakan sebagai berikut (Shafira, 2018) :

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (11)$$

Selain itu akurasi, nilai *precision* dan *recall* juga dapat digunakan untuk mengetahui kinerja pada model klasifikasi. *Precision/* presisi merupakan ukuran ketepatan berupa persentase dari *tupple* yang diklasifikasikan ke kelas positif yang benar-benar merupakan kelas positif. Proses presisi menggambarkan seberapa tepat suatu model dalam memprediksi kejaidian positif dalam serangkaian kegiatan prediksi. Untuk menghitung tingkat presisi dalam prediksi suatu kejadian dapat digunakan persamaan (Shafira, 2018) sebagai berikut:

$$Presision = \frac{TP}{TP+FP} \quad (12)$$

Sedangkan *recall* merupakan suatu ukuran kelengkapan berupa persentase *tupple* positif yang diklasifikasikan sebagai kelas positif. Perhitungan *recall* dilakukan berdasarkan persamaan (Shafira, 2018) berikut:

$$Recall = \frac{TP}{TP+FN} \quad (13)$$

Sedangkan untuk mengukur performa kelas minoritas digunakan *F-Measure. F-measure* merupakan perhitungan evaluasi dalam dalam informasi retrieval (temu kembali) yang mengkombinasikan *recall* dan *precision. F-Measure* dapat dihitung dengan rumus (Azis, *et al*. 2020) sebagai berikut:

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (14)$$

2.12 *Package* Keras

Keras merupakan salah satu *package* yang dapat digunakan dalam menyelesaiakan permasalahan mengenai jaringan syaraf. Pada dasarnya *package* keras dikembangkan dan memiliki fokus dalam mempercepat eksperimen pada proses konvolusi dan *recurrent* pada *neural networks*, maupun kombinasi antar keduanya. Dalam pembuatan model jaringan syaraf dengan menggunakan keras, kita tidak perlu menuliskan kode untuk menuliskan perhitungan matematisnya satu persatu. Hal ini disebabkan karena keras telah menyediakan beberapa model dasar yang dapat digunakan dalam optimasi mempermudah penelitian tentang *deep learning* atau CNN. Dalam proses komputasi dengan menggunakan keras akan berjalan dengan lancar dengan menggunakan CPU ataupun GPU (Shafira, 2018).