BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian ini dirancang dengan menerapkan algoritma CNN dalam mengklasifikasikan penyakit pada tanaman jagung berdasarkan citra daun. Studi sebelumnya mengenai klasifikasi menggunakan CNN dijadikan sebagai dasar, mengingat metode tersebut terbukti efektif untuk klasifikasi citra.

Salah satu penelitian terkait pada tahun 2023 [9] mengenai sistem identifikasi penyakit pada tanaman jagung menggunakan metode CNN dengan arsitektur *Alexnet* untuk mengklasifikasikan jenis penyakit tanaman jagung. Data akan diolah melalui beberapa tahap. Dataset yang digunakan pada penelitian ini berupa tiga kelas penyakit tanaman jagung yaitu, *blight, common rust* dan *grey leaf spot* dan satu jenis tanaman sehat dengan total keseluruhan data 4188 dataset. Dataset tersebut dapat diakses *online* melalui website *kaggle*. Parameter yang diujikan pada penelitian ini yaitu *optimizer, learning rate,* jumlah *epochs, input size,* dan *batch size* berpengaruh terhadap performa sistem yang berupa nilai akurasi, *precision, recall, fl-score,* dan *loss*. Sistem mampu mengklasifikasi dengan benar sebanyak 754 data dengan rincian dapat memprediksi penyakit *blight* pada tanaman padi sebanyak 203 data, penyakit *common rust* sebanyak 274 data, penyakit *grey leaf spot* sebanyak 67 data, dan tanaman jagung sehat sebanyak 198 data.dari total keseluruhan data 1047 data. Penelitian ini didapatkan hasil terbaik dengan penggunaan *optimizer* SGD, *learning rate* 0,01, jumlah *epochs* 20, *input size* 128x128, dan *batch size* 32 didapatkan hasil performa sistem dengan nilai akurasi, *precision, recall, fl-score,* dan *loss* masing-masing sebesar 89%, 87%, 85%, 85% dan 0,2852, serta grafik *performa* akurasi dan *loss* secara *good fit.*

Penelitian terkait pada tahun 2023 [10] mengenai identifikasi gambar daun jagung metode CNN dengan arsitektur *EfficientNet-B0* dan *Resnet-50*. Dataset yang diterapkan dalam penelitian ini terdiri dari 4 kelas, yakni *Blight, Common Rust, Gray Leaf Spot,* dan Tanaman Sehat. Metode CNN diterapkan dalam penelitian ini

dengan menggunakan dua model berbeda, yaitu *EfficientNet-B0* dan *ResNet-50*. Dataset yang digunakan diambil dari *website Kaggle* yang berjudul kumpulan data penyakit daun jagung. Dataset ini mempunyai 4 klasifikasi dengan jumlah data citra sebanyak 4188 dengan rincian masing-masing kelas yaitu data citra *blight* 1146, data citra *common rust* 1306, data citra *gray leaf spot* 574, dan data citra *healthy* 1162. Model *EfficientNet-B0* memiliki total 230 lapisan. Sedangkan model *ResNet-50* memiliki total kedalaman lapisan 50. Model pada penelitian ini menggunakan model dasar dua lapis. Masing-masing menggunakan *Global Average Pooling* 2D, *Dense* (512), menggunakan *Dropout* (0.5) dan (0.1), *optimizer* adamax dan rmsprop, serta menggunakan aktivasi Relu. Hasil penelitian menunjukkan akurasi sebesar 94% untuk *EfficientNet-B0* dan 93% untuk *ResNet-50*.

Penelitian berikutnya pada tahun 2022 [11] mengenai identifikasi citra penyakit pada daun jagung menggunakan *Deep Learning*. Penelitian ini dengan *Deep Learning* yang menggunakan metode klasifikasi algoritma *Convolutional Neural Network* (CNN). Menggunakan citra fisik pada daun tanaman jagung, metode CNN dapat membuat klasifikasi melalui model yang dibuat. Peneliti membuat sebuah model untuk dilakukan klasifikasi dengan bagian terdiri dari 4 *convolution layer*, 2 *pooling layer* dengan ukuran 2×2, 3 *dropout layer*, 2 *dense layer* serta 1 *flatten layer*. Untuk melakukan aktivasi menggunakan ReLu, beserta 32 dan 64 filter menggunakan 4 macam ukuran *kernel* yakni 3x2, 3x3, 3x4, 4x4. Dan dilakukan pengujian dengan 900 data gambar yang di mana 720 digunakan sebagai data *train* dan 180 sebagai data *Test.* Dengan *learning rate* sebesar 0.004, 100 *epoch* serta 6 algoritma performansi sebagai perbandingan yakni algoritma *Root Mean Square Propagation* (RMSProp), *Adaptive Gradient* (AdaGrad), *Stochastic Gradient descent* (SGD), *Adaptive Moment* (Adam), Adamax, Adadelta. Dan dihasilkan tingkat akurasi tertinggi yang dihasilkan oleh ukuran *kernel* 3x3 dengan algoritma optimasi *Adaptive Moment* (Adam) dengan hasil tingkat akurasinya sebesar 84% untuk data *test* dan 89% untuk data *train*, pada pengujian *Testing* dilakukan dengan jumlah 180 data yang didapatkan hasil tertinggi dengan model ukuran *kernel* 3x3 dengan jumlah *true* 175 dan jumlah *false* 5 didapatkan nilai presisi yang dihasilkan sebesar 94%, berdasarkan dengan komposisi warna pada citra.

2.2 DASAR TEORI

2.2.1 Tanaman Jagung

Jagung (*Zea Mays L.*) merupakan jenis tanaman biji-bijian yang tergolong dalam keluarga *Poaceae* atau rumput-rumputan. Tanaman jagung memiliki karakteristik morfologi yang khas, seperti batang yang tegak, daun yang panjang dan lebar, serta tongkol bunga yang mengandung biji jagung. Buah jagung berbentuk tongkol dan berfungsi sebagai wadah bagi biji jagung. Biji jagung sendiri memiliki ciri khas dengan adanya lapisan luar yang dikenal sebagai kulit jagung. Adapun untuk tanaman jagung terdapat pada Gambar 2.1.

Gambar 2. 1. Tanaman Jagung [12]

Pada Gambar 2.1 adalah gambar dari tanaman jagung. Indonesia seperti beberapa negara di Asia Tenggara lainnya, mengandalkan jagung sebagai sumber makanan tambahan selain beras atau padi. Tanaman jagung memiliki variasi karakteristik yang berbeda-beda, seperti jagung manis yang sering dikonsumsi langsung, dan jagung yang digunakan sebagai bahan baku industri. Namun, pertumbuhan jagung tidak hanya ditentukan oleh penanaman dan panen, tetapi juga perlu diawasi untuk memastikan kondisi kesehatannya. Jika daun jagung menunjukkan warna hijau segar tanpa adanya bercak atau tanda-tanda yang tidak normal, kemungkinan besar jagung tersebut sehat [12].

2.2.2 Jenis Penyakit Daun Jagung

Penyakit tanaman jagung adalah hasil dari interaksi 3 faktor utama yaitu patogen, inang, dan lingkungan. Penyebaran penyakit, baik dalam peningkatan intensitas maupun luasnya, sangat dipengaruhi oleh kontribusi signifikan dari ketiga faktor tersebut, yang akhirnya dapat mengakibatkan penurunan hasil. Terdapat 1 daun yang sehat dan 3 jenis penyakit pada daun jagung, yaitu

a. Daun sehat (*Healthy*)

Daun jagung yang sehat adalah daun yang berada dalam kondisi baik, tanpa adanya bercak atau perubahan warna pada permukaan daun. Adapun daun jagung sehat (*healthy*) terdapat pada Gambar 2.2.

Gambar 2. 2. Daun Jagung Sehat (*Healthy*) [13]

b. Daun hawar (*Blight)*

Hawar daun merupakan penyakit tanaman jagung di berbagai negara, termasuk Amerika, Asia, Afrika, dan Eropa. Penyakit ini dapat menyebabkan kerugian hingga mencapai 50%. Gejala awal penyakit hawar jagung melibatkan infeksi pada daun dengan tanda berupa bercak-bercak kecil yang berbentuk lonjong. Seiring perkembangannya, bercak tersebut akan menyebar dan berubah menjadi nekrotik. Adapun untuk daun jagung hawar (*blight*) terdapat pada Gambar 2.3.

Gambar 2. 3. Daun Jagung Hawar (*Blight*) [13]

c. Penyakit daun yang berkarat (*common rust*)

Karat daun disebabkan oleh jamur *Puccinia sorgii*, yang dapat teridentifikasi melalui gejala berupa bintik merah dan pelepasan partikel bubuk berwarna coklat atau kuning. Penyakit daun ini menghambat proses fotosintesis tanaman, menyebabkan pertumbuhan tanaman menjadi lebih lambat, dan berpotensi menyebabkan kematian pada tanaman. Adapun untuk daun jagung karat (*common rust*) terdapat pada Gambar 2.4.

Gambar 2. 4. Karat Daun (*Common Rust*) [13]

d. Bercak daun (*gray leaf spot*)

Penyakit jamur pada daun yang dikenal sebagai *Pyricularia grisea* dikenal dengan istilah bercak daun abu-abu. Gejala awal dari bercak daun abu-abu melibatkan lesi bulat kecil dengan cincin kuning di sekeliling daun. Lesi awal ini mungkin mengalami perubahan warna menjadi cokelat atau kecoklatan sebelum jamur mulai menghasilkan spora. Pada tahap ini, mengidentifikasi penyakit asli, yang berupa bercak daun berwarna abu-abu, dapat menjadi sulit karena kemiripannya dengan bercak karat atau bercak mata [14]. Adapun untuk daun jagung bercak (*gray leaf spot*) terdapat pada Gambar 2.5.

Gambar 2. 5. Bercak Daun (*Gray Leaf Spot*) [13]

2.2.3 *Google Colab*

Google Colab, yang juga dikenal sebagai *Google Colaboratory* adalah suatu *platform* pengkodean Python yang disusun dalam format "*notebook*". Dengan kata lain, Google menyediakan struktur komputasi yang memungkinkan pengguna untuk membuat program atau melakukan analisis data menggunakan Python. Secara khusus, *Google Colab* memberikan akses ke sumber daya *cloud* berbasis *runtime*, yang dapat diakses melalui browser seperti Chrome, Firefox, dan Safari. Selain itu, terdapat tiga jenis prosesor yang dapat dimanfaatkan, yaitu *Central Processing Unit* (CPU), *Graphic Processing Unit* (GPU), dan *Tensor Processing Unit* (TPU), sesuai dengan kebutuhan pengguna dalam menjalankan tugas-tugas *machine learning*.

Platform ini juga menciptakan suatu lingkungan pembelajaran Python yang bersifat *open source* [15]. Adapun untuk logo *google colab* terdapat pada Gambar 2.6.

Gambar 2. 6. Logo *Google Colab* [15]

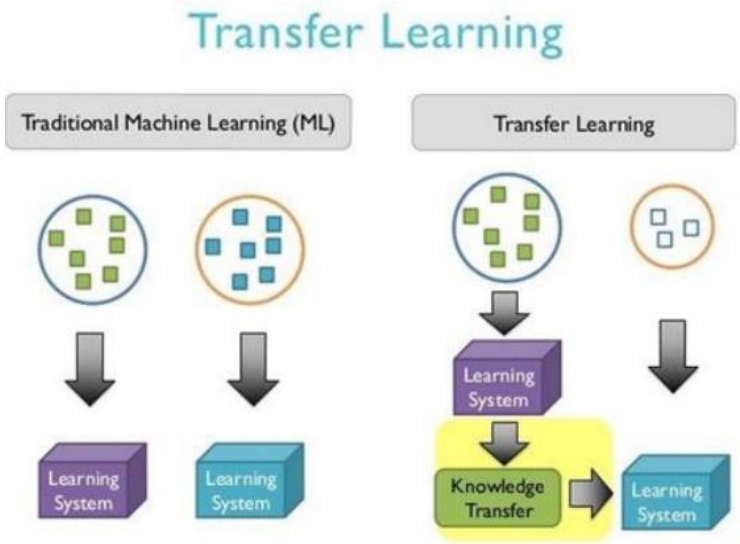
2.2.4 *Python*

Python merupakan bahasa pemrograman yang bersifat interpretatif dan sangat serbaguna, didasarkan pada filosofi desain yang menekankan keterbacaan kode. Diklaim sebagai bahasa yang menggabungkan kapabilitas dan kemampuan, *Python* mencolok dengan sintaksis kode yang sangat jelas. *Python* menonjolkan efisiensi waktu, kemudahan pengembangan, dan kompatibilitas dengan berbagai sistem. Penggunaan *Python* tidak hanya terbatas pada pembuatan aplikasi *standalone*, namun juga dapat digunakan untuk pemrograman *skrip*. Sebagai bahasa pemrograman dinamis, *Python* mendukung pemrograman berbasis objek. [16]. *Python* dapat diakses dan digunakan secara bebas untuk berbagai tujuan, termasuk keperluan komersial, dengan tersedia dalam berbagai lisensi dari beberapa versi. Adapun untuk logo *python* terdapat pada Gambar 2.7.

Gambar 2. 7. Logo *Python* [16]

2.2.5 *Transfer Learning*

Transfer learning adalah metode dalam *machine learning* yang menggunakan model yang telah dilatih sebelumnya sebagai dasar untuk melatih model baru. Pendekatan *Transfer Learning* melibatkan penggunaan pembelajaran yang sudah ada dan diterapkan kembali ke awal untuk model baru yang difokuskan pada tugas khusus. Keuntungan utama dari *Transfer Learning* terletak pada kemampuannya untuk mengatasi masalah *overfitting*, yang muncul ketika model terlalu beradaptasi dengan data pelatihan dan kurang efektif dalam generalisasi pada data baru. Pelatihan model dari awal memakan waktu dan sumber daya yang signifikan. Dengan *Transfer Learning*, model dapat memanfaatkan pengetahuan dari model yang sudah dilatih sebelumnya, mengurangi waktu pelatihan dan meningkatkan efisiensi. Selain itu, *Transfer Learning* dapat meningkatkan kinerja model dalam tugas yang berbeda, karena model yang telah dilatih pada tugas tertentu dapat memberikan wawasan berharga untuk tugas yang baru. Sebagai contoh, model *pre-trained* yang awalnya digunakan untuk mengenali mobil dapat diterapkan dengan sukses untuk mengenali truk. Adapun untuk *transfer learning* terdapat pada Gambar 2.8.



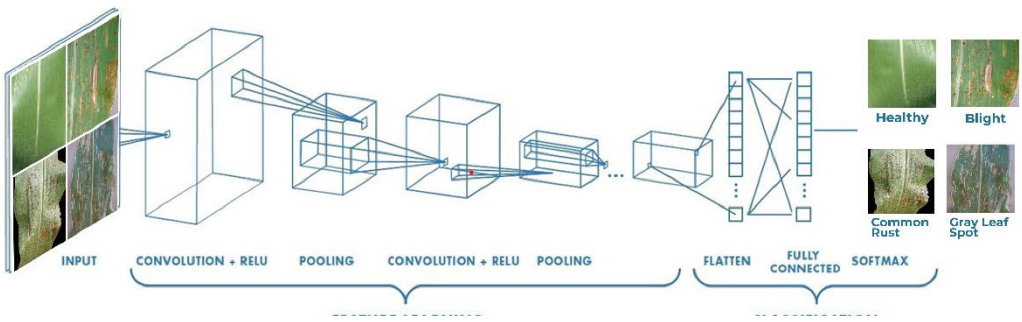
Gambar 2. 8. *Transfer Learning* [17]

Pada *transfer learning* menggunakan suatu pengetahun (*knowledge*) pada suatu *task* T1, untuk menyelesaikan permasalahan *task* T2 . Disumsi bahwa T1 memiliki kaitan dengan T2, sedemikian sehingga fasih pada T1 akan menyebabkan

kita fasih pada T2 (atau lebih fasih dibandingan tidak menguasai T1 sama sekali). Perhatikan Gambar 2.8 yang mengilustrasikan perbedaan pembelajaran mesin biasa dan penggunaan *transfer learning*. Pada pembelajaran mesin biasa, kita melatih model untuk masing-masing *task*. Pada *transfer learning* dapat menggunakan model yang sudah ada, disebut *pre trained model*, untuk *task* baru. Selain dimotivasi oleh kemiripan kedua *tasks*, *transfer learning* juga dimotivasi oleh ketersediaan data. Misal dataset untuk *task* T1 banyak, sedangkan untuk *task* T2 sedikit. Berhubung T1 dan T2 memiliki kemiripan, model untuk T1 yang diadaptasi untuk T2 akan *konvergen* lebih cepat dibanding melatih model dari awal untuk T2 [17].

2.2.6 *Convolutional Neural Network* (CNN)

CNN merupakan jenis *neural network* yang umumnya digunakan dalam pengolahan data citra dan merupakan salah satu metode *neural network* yang paling terkenal dan diminati. CNN memiliki kemampuan untuk memproses data berdimensi tinggi seperti video dan gambar. Prinsip kerja CNN mirip dengan *neural network* pada umumnya, dengan perbedaan utama menggunakan *kernel* 2 dimensi atau dimensi tinggi pada setiap unit dalam lapisan CNN untuk menjalankan proses konvolusi.. Selanjutnya, CNN menggunakan berbagai parameter untuk mengurangi jumlah parameter agar lebih mudah dipelajari. *Output* dari lapisan konvolusi disebut sebagai *feature map* yang mendeskripsikan ciri-ciri unik dari citra. Lapisan konvolusi terdiri dari *filter* yang berfungsi melakukan proses konvolusi pada matriks citra masukan. CNN melibatkan beberapa lapisan, antara lain *convolution layer*, fungsi *activation layer*, *pooling layer*, dan *fully connected layer* [18]. Ilustrasi CNN terdapat pada Gambar 2.9



Gambar 2. 9. Pemodelan CNN [18]

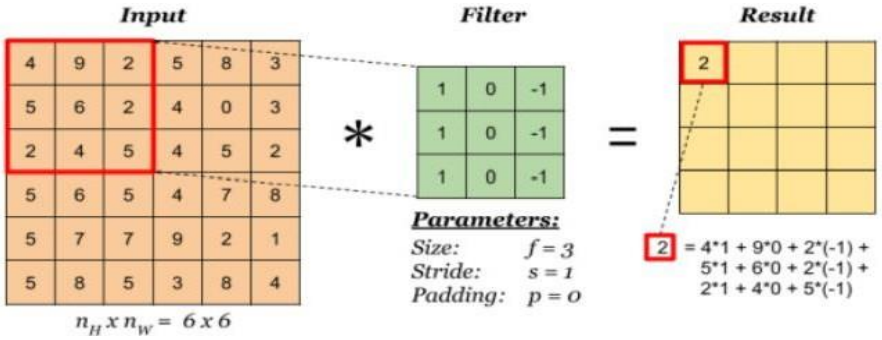
Berdasarkan Gambar 2.9 mengenai pemodelan CNN berikut penjelasannya, adalah:

1. *Feature Learning/ Feature Extration Layer*

   Feature Extraction Layer merupakan tahap dimana citra di ubah menjadi fitur-fitur numerik proses yang dapat merepresentasikan citra tersebut.

   a. *Convolutional Layer*

   Tujuan proses konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input*. Konvolusi menghasilkan transformasi *linear* dari data *input* sesuai informasi *spasia*l dalam data tersebut. Bobot pada *layer* ini menentukan *kernel* konvolusi yang dapat diatur berdasarkan *input* pada CNN. Dalam proses ini, *Convolution layer* menggunakan *filter* pada setiap citra masukan. Filter ini berupa matriks 2 dimensi dengan ukuran 5×5, 3×3 atau 1×1. Proses *Convolution layer* dengan menggunakan *filter* akan menghasilkan *feature map* yang kemudian akan ditambahkan *activation function*. Adapun Proses operasi *convolutional layer* terdapat pada Gambar 2.10.



Gambar 2. 10. Operasi *Convolutional Layer* [18]

Berdasarkan Gambar 2.10 menjelaskan dalam pengolahan gambar, konvolusi merujuk pada penerapan *filter* (hijau) ke seluruh *input*. Parameter-filter yang digunakan mencakup ukuran *filter* (f) sebesar 3 x 3 pergeseran filter (s) sebanyak 1, dan *padding* (p) dengan jumlah 0. Nilai dalam filter berkisar antara -1 hingga 1, dan nilai ini digunakan untuk menentukan hasil konvolusi, seperti efek ketajaman, blur, *grayscale*, *sharpen,* dan sebagainya. Bidang *input* (merah) adalah gambar yang akan mengalami konvolusi, di mana angka dalam setiap *piksel* mencerminkan matriks dari gambar tersebut. Pada kotak kuning menunjukkan hasil konvolusi dari gambar tersebut. Proses konvolusi pada data

citra bertujuan untuk mengekstrak fitur dari citra masukan [19]. Berikut penjelasan dari stride dan padding, yaitu :

a) Stride

Stride merupakan parameter yang menentukan seberapa jauh *filter* atau *kernel* bergerak ketika melakukan operasi konvolusi pada data *input*. Apabila nilai *stride* adalah 1, *filter* akan bergeser satu piksel pada setiap langkahnya, baik secara *horizontal* maupun *vertikal*. Pemilihan nilai *stride* yang sesuai berpengaruh pada dimensi *output* dari lapisan konvolusi dan dapat memengaruhi tingkat kompleksitas komputasi. Penggunaan *stride* yang lebih besar cenderung menghasilkan representasi yang lebih kasar dari data *input*, sementara *stride* yang lebih kecil dapat memberikan representasi yang lebih rinci.

b) Padding

Padding atau *Zero Padding* adalah parameter yang menentukan jumlah piksel yang ditambahkan pada setiap sisi *input*, biasanya diisi dengan nilai 0. Penambahan ini bertujuan untuk mengubah dimensi *output* dari lapisan konvolusi (*Feature Map*). Fungsi dari *padding* adalah memastikan bahwa *output* dari lapisan konvolusi selalu memiliki setidaknya dimensi yang sama dengan *input*. Penggunaan *padding* bermanfaat agar *output* dari lapisan konvolusi tidak mengalami pengurangan dimensi yang signifikan dibandingkan dengan *input*. Hal ini penting karena *output* dari lapisan konvolusi tersebut digunakan sebagai *input* untuk lapisan konvolusi berikutnya, sehingga penggunaan *padding* membantu meminimalkan kehilangan informasi. Dengan adanya *padding*, dimensi *output* dapat diatur untuk tetap sama dengan *input* atau setidaknya tidak mengalami penurunan yang drastis dari *input*. Ini memungkinkan penggunaan lapisan konvolusi yang lebih dalam, sehingga fitur-fitur yang diekstraksi menjadi lebih beragam. Sebagai contoh, jika *input* memiliki dimensi 5x5 dan dilakukan konvolusi dengan *filter* 3x3 serta *stride* sebesar 2, maka *feature map*

yang dihasilkan berukuran 2x2. Namun, dengan penambahan *zero padding* sebanyak 1 *piksel* di setiap sisi, *feature map* dapat diperluas menjadi ukuran 3x3, yang menghasilkan lebih banyak informasi. Untuk menghitung dimensi dari *feature map* bisa menggunakan rumus seperti dibawah ini [19]:

$$Output = \frac{W-N+2P}{S} + 1 \quad (1)$$

Dimana :

W = Panjang/Tinggi *Input*

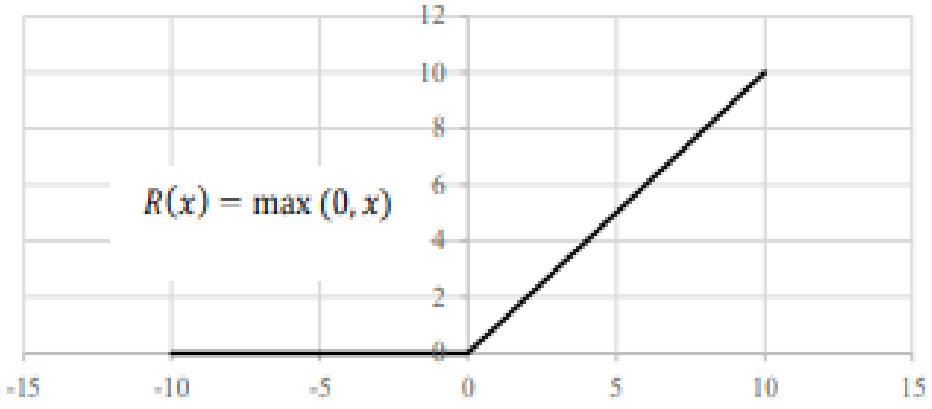
N = Panjang/Tinggi *Filter*

P = *Zero Padding*

S = *Stride*

b. Rectified Linear Unit (ReLU)

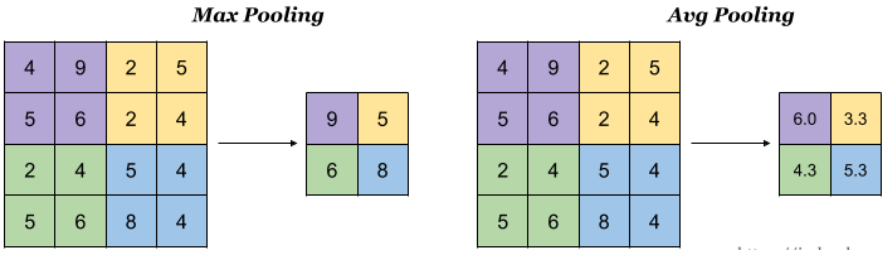
ReLU digunakan sebagai fungsi aktivasi pada lapisan konvolusi untuk mengatur nilai keluaran proses konvolusi. Fungsi ini bertujuan untuk mengubah nilai x jika menjadi 0 (nol) jika nilainya negatif. Namun, jika *input* positif atau tidak kurang dari 0, maka *output* akan tetap sama. Adapun fungsi aktivasi RELU terdapat pada Gambar 2.11.



Gambar 2. 11. Fungsi Aktivasi RELU [18]

c. *Pooling Layer*

Pooling Layer memiliki peran dalam menerima *input* dari fungsi aktivasi dan mengurangi jumlah parameter. *Layer* ini sering disebut sebagai *subsampling* atau *downsampling*, yang mengurangi dimensi dari *feature map* tanpa kehilangan informasi krusial di dalamnya. Dua jenis *pooling layer* yang umum digunakan adalah *Max Pooling* dan *Average Pooling.* Adapun untuk proses *pooling layer* terdapat pada Gambar 2.12.



Gambar 2. 12. *Pooling Layer* [19]

Berdasarkan Gambar 2.12 mengenai proses *pooling layer* yang terdapat 2 yaitu *max pooling* dan *average pooling. Max pooling* mendapatkan nilai maksimum dari matriks *input* berdasarkan hasil operasi '*dot'* dari *convolution layer*. Sedangkan *average pooling* mengambil nilai rata-rata dari matriks hasil operasi konvolusi [20].

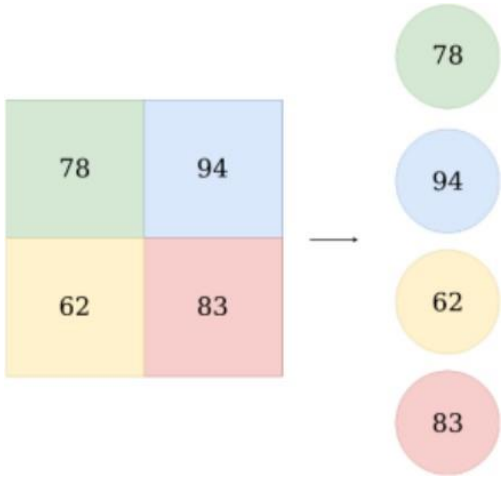
2. Lapisan Klasifikasi

Lapisan ini terbentuk dari sejumlah lapisan yang berisi *neuron* yang terhubung sepenuhnya (*fully connected*) dengan lapisan lainnya. Lapisan ini menerima *input* dari *output layer* pada bagian pembelajaran fitur yang kemudian di proses dengan mengaplikasikan proses *flatten* dan menambahkan beberapa *hidden layer* pada proses *fully connected* untuk menghasilkan *output* berupa akurasi klasifikasi dari setiap kelas.

a. *Flatten*

Matriks hasil konvolusi dan *max pooling* akan melalui proses *flatten* sebelum masuk ke dalam *layer fully connected* dimana setiap citra akan diubah menjadi vektor. Hasil keluaran dari proses *flatten* akan menjadi masukan pada

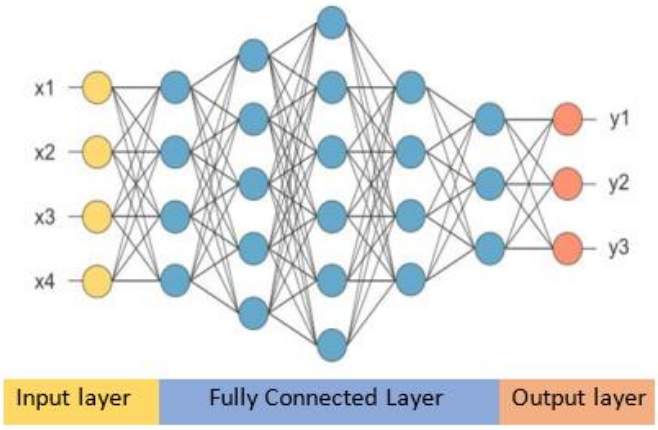
proses *fully connected layer*. Proses perubahan matriks menjadi *vector* diterapkan agar sesuai dengan format masukan pada *layer neural network*. Adapun untuk proses *flatten* terdapat pada Gambar 2.13.



Gambar 2. 13. Proses *Flatten* [20]

b. Fully connected layer

Lapisan *fully connected layer* adalah lapisan dalam jaringan saraf tiruan yang memiliki *neuron* aktivasi yang terkoneksi satu sama lain dari lapisan sebelumnya. *Neuron-neuron* ini dikonversi menjadi bentuk satu dimensi sebelum dihubungkan ke semua *neuron* pada lapisan sepenuhnya terhubung. Proses lapisan sepenuhnya terhubung ini dilakukan untuk memproses data dengan tujuan klasifikasi. Adapun untuk proses *fully connected layer* terdapat pada Gambar 2.14.



Gambar 2. 14. Proses *Fully Connected Layer* [21]

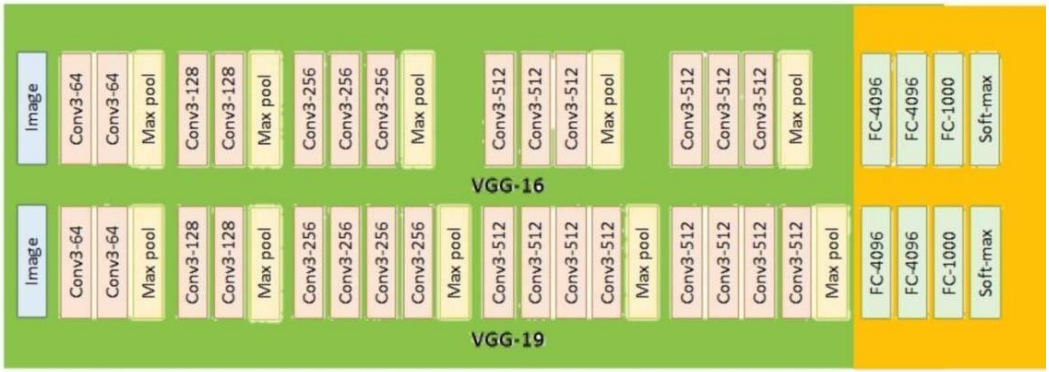
Pada Gambar 2.14 menjelaskan mengenai proses *fully connected layer*. Pada proses ini berbeda dengan *neuron* pada lapisan konvolusional yang hanya terhubung dengan suatu area tertentu dari input, neuron pada lapisan sepenuhnya terhubung terkoneksi secara menyeluruh.

c. Softmax

Softmax digunakan untuk menghitung probabilitas setiap kelas target terhadap semua kelas target. Fungsi ini berguna dalam menentukan kelas target yang sesuai untuk *input* yang diberikan [21].

2.2.7 Arsitektur VGG

Visual Geometry Group (VGG) adalah salah satu model arsitektur yang diterapkan dalam CNN untuk pengolahan citra. Model ini dikembangkan oleh Karen Simonyan dan Andrew Zisserman. Adapun untuk arsitektur VGG16 dan VGG19 terdapat pada Gambar 2.15.



Gambar 2. 15. Arsitektur VGG16 dan VGG19 [22]

Gambar 2.15 mengenai arsitektur dari VGG16 dan VGG19. Perbedaan dari VGG16 dan VGG19 ini terdapat pada jumlah lapisannya. Pada VGG16 terdiri dari total 16 *layers*, dengan 13 *layers* konvolusi dan 3 *layers fully connected*. Keunggulan VGG16 terletak pada homogenitas arsitekturnya yang secara konsisten menerapkan konvolusi 3x3 dan *pooling* 2x2 dari awal hingga akhir. Sedangkan pada VGG19 mengadopsi pendekatan bahwa semakin banyak lapisan, semakin baik akurasi model. Dengan 19 *layers*, termasuk 16 *layers* konvolusi dan 3 *layers*

fully connected, VGG19 menyediakan representasi gambar yang lebih kompleks, mengurangi potensi kesalahan dalam pemrosesan gambar menggunakan CNN [22].

2.2.8 Optimasi

Dalam konteks *machine learning*, optimasi mengacu pada proses penyesuaian parameter suatu model dengan tujuan meminimalkan atau memaksimalkan fungsi tujuan tertentu. Fungsi tujuan ini mengevaluasi kinerja model pada suatu tugas dan parameter merupakan variabel internal yang model gunakan untuk membuat prediksi. Proses optimasi melibatkan pencarian nilai-nilai parameter yang menghasilkan kinerja terbaik pada tugas yang diberikan. *Optimizer* adalah algoritma atau metode yang digunakan untuk memperbaharui parameter selama pelatihan model. Pilihan *optimizer* dapat secara signifikan memengaruhi kecepatan konvergensi dan kualitas akhir model. Beberapa *optimizer* umum termasuk SGD, Adam, RMSprop, dan lainnya. Setiap *optimizer* memiliki kelebihan dan kekurangan, dan pilihan *optimizer* dapat bergantung pada karakteristik dataset dan sifat tugas pembelajaran.

Dalam penelitian ini, optimasi yang digunakan adalah *stochastic gradient descent* (SGD). SGD adalah salah satu algoritma optimasi yang digunakan dalam *machine learning* untuk menemukan nilai minimum dari fungsi biaya. SGD digunakan untuk memperbarui bobot model secara iteratif dengan menghitung *gradien* dari fungsi biaya pada setiap sampel data yang dipilih secara acak [23].

Rumus atau formula SGD adalah:

$$\theta = \theta - \alpha * \nabla J(\theta;x(i);y(i)) \quad (2)$$

Dimana:

θ adalah vektor bobot model

α adalah *learning rate*, yaitu besarnya langkah yang diambil pada setiap iterasi

$\nabla J(\theta;x(i);y(i))$ adalah gradien dari fungsi biaya J terhadap vektor bobot θ pada sampel data ke-i dengan fitur x(i) dan label y(i)

2.2.9 *Confusion Matrix*

Confusion matrix merupakan metode evaluasi yang digunakan untuk mengukur kinerja pada tahap klasifikasi [23]. *Confusion matrix* berisi informasi tentang jumlah klasifikasi berhasil diprediksi dengan benar dan salah oleh sistem. Adapun untuk *confusion matrik* yang terdapat pada Tabel 2.1.

Tabel 2. 1. *Confusion Matrix* 3 Kelas [25]

Confusion Matrix		Predicted			False Negative (FN)	Recall
		Class 1	Class 2	Class 3		
Actual	Class 1	A	B	C	B + C	A/(A + B + C)
	Class 2	D	E	F	D + F	E/(D + E + F)
	Class 3	G	H	I	G + H	I/(G + H + I)
	False Positive (FP)	D + G	B+H	C+F	Overall Accuracy = A + E + I/(Sum of red and green squares)	
	Precision	A/(A + D + G)	E/(B + E + H)	I/(C + F + I)		

True positives True Negatives Misclassified cases False Positives False Negatives.

Tabel 2.1 menunjukkan bahwa setiap dari tiga kelas, yang diberi label "1", "2", dan "3", menunjukkan perbedaan nilai untuk FP dan FN. Secara spesifik, nilai FP dan FN untuk kelas "1" dapat dibedakan secara visual melalui penggunaan warna biru muda dan ungu. Demikian pula, TP dan TN untuk kelas "1" ditandai oleh kotak berwarna hijau muda dan hijau tua. Dengan memanfaatkan perhitungan TN, TP, FP, dan FN, kemudian menghitung matriks kinerja tambahan untuk setiap kelas secara berurutan [25].

Parameter evaluasi digunakan sebagai nilai perbandingan dari masing-masing model untuk mendapatkan model sistem yang optimal. Nilai-nilai tersebut akan dihasilkan dari *Confusion Matrix* dengan menggunakan parameter uji sebagai yaitu:

a. Akurasi (*Accuracy*)

Akurasi adalah rasio prediksi benar (positif dan negatif) terhadap keseluruhan data. Nilai dari akurasi yang diperoleh dari rumus yaitu [24]

$$\text{Akurasi} = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (3)$$

Dimana:

TP = *true positif*

TN = *true negative*

FP = *false positif*

FN = *false negative*

b. Presisi (*Precision*)

Presisi adalah rasio prediksi benar positif terhadap hasil prediksi positif secara keseluruhan. Rumus dari presisi, yaitu [24]:

$$\text{Presisi} = \frac{(\text{TP})}{(\text{TP+FP})} \tag{4}$$

Dimana :

TP = *true positif*

FP = *false positif*

c. Recall

Recall adalah rasio prediksi positif benar terhadap data positif benar dan negatif salah. Rumus dari *recall*, yaitu [24]:

$$Recall = \frac{(\text{TP})}{(\text{TP+FN})} \tag{5}$$

Dimana :

TP = *true positif*

FN = *false negative*

d. F1- Score

F1-Score adalah kalkulasi evaluasi yang menggabungkan hasil dari *recall* dan *presisi*. Pada situasi tertentu, ketika *recall* dan presisi dapat memiliki bobot yang berbeda [25]. Rumus *F1-Score*, yaitu [24]:

$$F1\ Score = 2\ \times \frac{(\text{Recall} \times \text{Presisi})}{(\text{Recall+Presisi})} \times 100 \tag{6}$$