BAB II. LANDASAN TEORI

2.1 Studi Literatur

Pada tahun 2020, terdapat penelitian (Prasmatio dkk., 2020) deteksi pengenalan ikan menggunakan *Convolutional Neural Network* (CNN). Tujuan dari penelitian ini adalah mengidentifikasi jenis ikan secara terotomasi tanpa harus bertemu dengan pakar secara langsung. Hasil yang didapatkan adalah dengan pengujian algoritma menghasilkan rata-rata pengujian sebesar 79,52% dari total 27 citra uji coba dengan menggunakan foto ikan. Akurasi dalam penelitian ini mendapatkan nilai sebesar 85,14% dari hasil 23 dapat memprediksi dengan benar dan 4 tidak dapat diprediksi dengan benar. Selanjutnya mendapatkan nilai presisi sebesar 77,8% dan nilai *recall* sebesar 85,2%.

Penelitian yang dilakukan oleh (Fauzi dkk., 2019) melakukan penelitian untuk mengimplementasi *Convolutional Neural Network* untuk identifikasi ikan air tawar. Penelitian ini dilakukan untuk mengimplementasikan algoritma CNN untuk identifikasi ikan air tawar serta mengukur tingkat akurasi hasil klasifikasi. Tingkat akurasi dari model klasifikasi yang dibuat peneliti dapat ditingkatkan dengan berbagai cara diantaranya dengan menambahkan lapisan pada *layer* CNN, menambahkan *dataset* agar lebih banyak, serta melakukan segmentasi citra terlebih dahulu sebelum melakukan proses klasifikasi menggunakan CNN. Model klasifikasi CNN yang dibuat oleh peneliti mampu mengenali data citra ikan air tawar dengan tingkat akurasi yang baik yaitu 88,33%.

Penelitian lain yang dilakukan (Ahmadi, 2022) mengestimasi panjang ikan kerapu hidup menggunakan koreksi *de-haze* dan metode *Deep Learning*. Penelitian ini bertujuan untuk mengaplikasikan algoritma koreksi gambar *de-haze* dan kecerdasan buata (*deep learning*) untuk mengestimasi panjang ikan kerapu hidup. Model yang dibuat oleh peneliti menghasilkan nilai akurasi sebesar 92.00%. Nilai *precision* pada model ini mencapai 92.00%. Nilai *recall* pada model ini mampu mencapai 100%.

Berikut ini merupakan beberapa penelitian terdahulu yang relevan dengan pengembangan sistem *Grading* Bibit Ikan Lele Dengan Metode CNN:

Tabel 2.1 Studi Literatur

No	Penulis	Judul	Metode	Kesimpulan	Tahun
1	R. Mehindra Prasmatio, Basuki Rahmat, dan Intan Yuniar	Deteksi dan Pengenalan Ikan Menggunakan Algoritma *Convolutional Neural Network*	CNN	Penelitian ini menghasilkan tingkat akurasi sebesar 85,15%	2020
2	Septian Fauzi, Puspa Eosina, dan Githa Fitri Laxmi	Implementasi *Convolutional Neural Network* untuk Identifikasi Ikan Air Tawar	CNN	Model klasifikasi CNN yang dibuat oleh peneliti mampu mengenali data citra ikan air tawar dengan tingkat akurasi yang baik yaitu 88,33%	2019
3	Yudhi Ahmadi	Estimasi Panjang Ikan Kerapu Hidup di Keramba Jaring Apung Menggunakan Koreksi Gamabr *De-Haze* dan Metode Deep Learning	*De-Haze*, *Deep Learning*	Penelitian ini mendapatkan nilai akurasi sebesar 92.00%, *presicison* sebesar 92.00%, *recall* sebesar 100%, dan F1-*Score* 95,83%	2022
4	Rizal Fachmi, Achmad Hidayanto, dan Yosua Alvin Adi Sutrisno	Sistem Identifikasi Ukuran Tubuh Menggunakan Metode *Convolutional Neural Network* (CNN)	(CNN), Regresi Linier, polynomial kudratik	Penelitian ini menyarankan untuk melakukan penelitian lebih dalam lagi karena adanya perbedaan nilai aktual dan estimasi	2020
5	Aji Pangestu dan Aries Muslim	*Deep Learning* untuk Mendeteksi Pola Kebotakan	CNN	Hasil akurasi sudah cukup mendapatkan hasil	2022

		Rambut pada Pria dengan Metode CNN (*Convolutional Neural Network*)		yang optimal dalam pendeteksianya, untuk akurasi pada openCV library hasil deteksi menunjukan 70% - 99% pada objek hasil deteksi.	
6	Denny Nur Ramadhan	Menuju Kacamata Cerdas Tunanetra: Pengenalan Wajah Menggunakan Framework Tensorflow Lite	CNN	Sistem yang dirancang mampu mengenali wajah dengan akurasi tertinggi sebesar 87% didalam ruangan pada jarak dekat seperti yang ditunjukan pada skenario 3, sedangkan apabila model diuji coba diluar ruangan akurasi menurun menjadi 78% pada jarak dekat.	2022

Berdasarkan studi literatur diatas, maka dilakukanlah penelitian pengembangan sistem *Grading* Bibit Ikan Lele Dengan Algoritma CNN. *Deep Learning* dapat belajar untuk melakukan klasifikasi secara langsung dari gambar dan suara (Ilahiyah & Nilogiri, 2018). CNN merupakan jenis algoritma yang baik dalam mengenali citra digital ikan air tawar (Fauzi dkk., 2019). CNN juga dapat mengidentifikasi ikan secara *real-time* yang terbukti efisien dalam klasifikasi ikan (Prasmatio dkk., 2020).

2.2 Dasar Teori

2.2.1 Ikan Lele

Ikan lele merupakan salah satu ikan air tawar yang sudah banyak dikenal di Indonesia. Banyak orang yang membudidayakan ikan lele karena ikan lele dapat hidup di kondisi air dan kadar oksigen yang rendah sehingga budidaya ikan lele tergolong mudah (Zuraidah dkk., 2021). Ikan lele memiliki ciri-ciri fisik tubuh yang licin, bentuknya memanjang, tidak memiliki sisik, sirip punggung dan sirip anus juga berbentuk memanjang. Sirip dan ekor lele terkadang menyatu hingga membuatnya tampak seperti sikat yang pendek. Hasil bididaya ikan lele umumnya digunakan sebagai konsumsi dan pembenihan ikan. Ikan lele dewasa digunakan sebagai pahan pangan dan benih ikan lele digunakan untuk pembenihan budidaya ikan lele (Ulva dkk., 2018).

2.2.2 Grading

Grading adalah kegiatan setelah sortasi yang dilakukan untuk mengelompokkan produk sesuai dengan bentuk, warna atau jenis yang dimiliki (Edowai & Tahoba, 2018). Pada dasarnya, *grading* digunakan untuk mengelompokkan produk atau bahan mentah ke dalam kategori yang berbeda sesuai dengan kualitas atau ukurannya. *Grading* juga dapat dilakukan dengan menggunakan alat-alat mekanis atau teknologi yang canggih, seperti sensor atau kamera. Alat-alat tersebut dapat digunakan untuk mengukur kualitas atau ukuran produk atau bahan mentah dengan cepat dan akurat. Proses *grading* bibit ikan lele merupakan proses yang krusial bagi peternak bibit ikan lele. Jika bibit ikan lele yang disortir tidak memiliki ukuran yang sesuai, maka hasil sortir yang didapatkan dapat menyebabkan kerugian pada peternak dan pembeli.

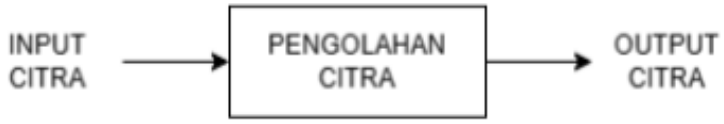
2.2.3 Citra Digital

Citra digital adalah gambar dua dimensi yang dihasilkan dari analog dua dimensi yang kontinu menjadi gambar melalui proses sampling. Gambar analog dibagi menjadi N baris dan M kolom sehingga menjadi gambar diskrit. Citra digital merupakan citra yang dapat diolah komputer. Yang disimpan dalam komputer hanyalah angka-angka yang menunjukkan besar intensitas pada masing-masing

piksel. Karena berbentuk data numerik, maka citra digital dapat diolah dengan komputer (Munantri dkk., 2020).

2.2.4 Pengolahan Citra Digital

Pengolahan citra digital adalah ilmu yang mempelajari hal-hal berkaitan dengan perbaikan kualitas terhadap suatu gambar (meningkatkan kontras, perubahan warna, restorasi citra), transformasi gambar (translasi, rotasi transformasi, skala, geometrik), melakukan pemilihan citra ciri (feature images) yang optimal untuk tujuan analisis, melakukan penyimpanan data yang sebelumnya dilakukan reduksi dan kompresi, transmisi data, dan waktu proses data. Adapun diagram sederhana dari proses pengolahan citra dapat dilihat pada gambar 2.1.



Gambar 2.1 Proses Pengolahan Citra

Sebuah citra digital dapat diwakili oleh sebuah matriks yang terdiri dari M kolom dan N baris, dimana perpotongan antara kolom dan baris tersebut disebut dengan pixel (picture element) yang merupakan elemen terkecil dari sebuah citra. Pixel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x, y) adalah f (x, y) yang merupakan besar intensitas atau warna dari pixel di titik itu (Munantri dkk., 2020).

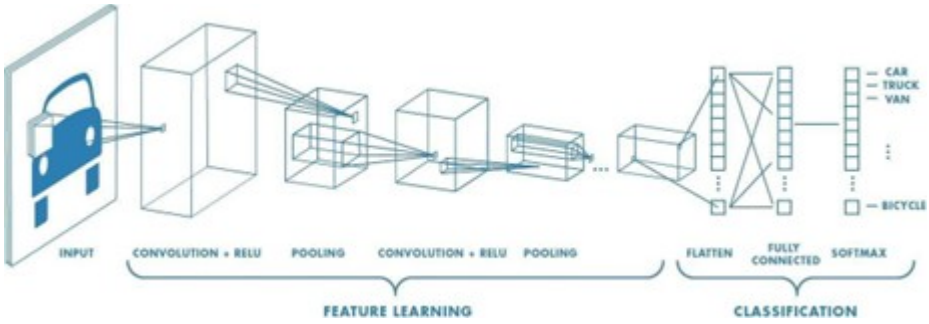
2.2.5 Deep Learning

Deep Learning termasuk dalam bidang Kecerdasan Buatan (AI), memanfaatkan beberapa *layer* pengolahan informasi *nonlinear* untuk ekstraksi fitur, pengenalan pola, dan klasifikasi (Deng & Yu, 2013). *Deep Learning* mampu belajar dalam melakukan klasifikasi secara langsung dari gambar atau suara (Ilahiyah & Nilogiri, 2018). *Deep Learning* menggunakan sistem algoritma berlapis yang disebut *Artificial Neural Network* (ANN). ANN dibuat berdasarkan konsep sistem kerja saraf otak manusia dimana neuron-neuron saraf otak saling terkoneksi satu sama lain sehingga membentuk sebuah jaringan neuron (Nugroho dkk., 2020).

2.2.6 Convolutional Neural Network

Metode Deep Learning yang saat ini memiliki hasil paling signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN). Hal tersebut dikarenakan CNN berusaha meniru sistem pengenalan citra pada visual cortex manusia sehingga memiliki kemampuan mengolah informasi citra (Nugroho dkk., 2020).

Convolutional Neural Network (CNN) merupakan pengembangan dari *Multilayer Perceptron* (MLP) yang termasuk dalam *neural network* bertipe *feed forward*. *Convolutional Neural Network* didesain untuk mengolah data dua dimensi. CNN memanfaatkan proses konvolusi dengan mengkalikan matriks dari kernel konvolusi *(filter)* pada citra yang didapatkan. Oleh karena itu, algoritma CNN dapat digunakan dalam pengenalan wajah, analisis dokumen, klasifikasi gambar, dan klasifikasi video. *CNN* termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Secara garis besar, CNN tidak terlalu jauh berbeda dengan *neural network* biasanya. CNN terdiri dari neuron yang memiliki *weight*, *bias* dan *activation function*. Arsitektur CNN dapat dilihat pada gambar 2.2.



Gambar 2.2 Arsitektur CNN

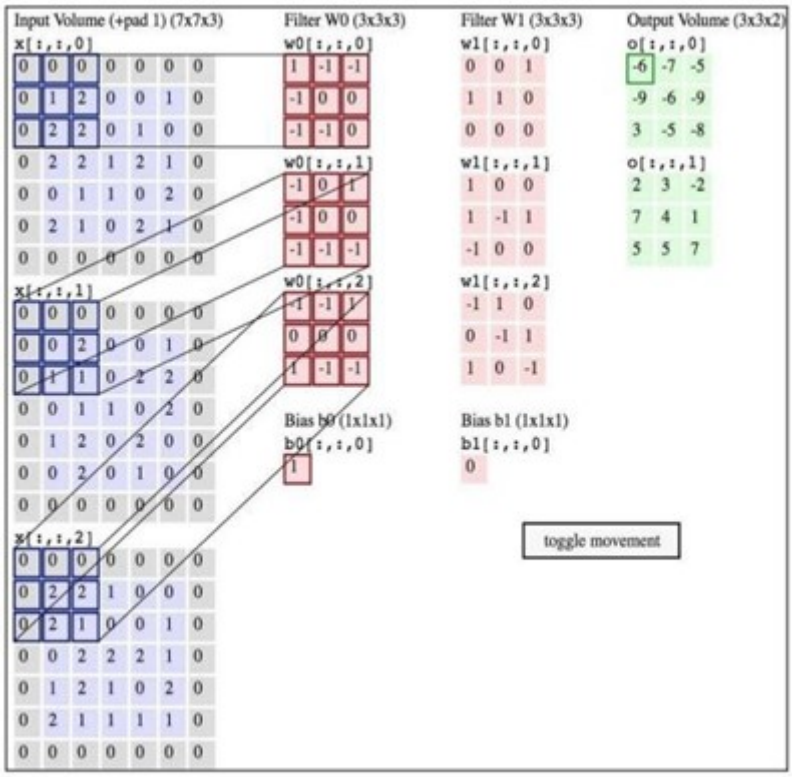
(Sumber: (Krizhevsky dkk., 2017))

Dari gambar 2.2 dapat dilihat Arsitektur dari CNN dibagi menjadi 2 bagian besar, *Feature Extraction Layer* dan *Fully-Connected Layer* (MLP):

1. *Feature Extraction Layer* terdiri dari dua bagian.
   - *Convolution Layer*

     Convolutional Layer merupakan *layer* yang pertama kali menerima gambar yang diinputkan. *Layer* ini melakukan proses konvolusi

menggunakan sebuah *filter*. *Filter* ini diinisialisasi dengan nilai tertentu (*random* atau menggunakan teknik tertentu seperti Glorot), dan nilai dari *filter* inilah yang menjadi parameter yang akan di-*update* dalam proses learning. *Filter* ini akan bergeser ke seluruh bagian gambar. Pergeseran tersebut akan menghasilkan dot product antara input dan nilai dari *filter* tersebut. Dengan menggeser *filter* keseluruh bagian gambar, dihasilkan sebuah *output* yang disebut sebagai *activation map* atau *feature map* (Fauzi dkk., 2019). Proses Konvolusi dapat dilihat pada gambar 2.3.



Gambar 2.3 Proses Konvolusi

(Sumber: (Orhan & Bastanlar, t.t.))

Pergeseran *filter* ditentukan oleh sebuah parameter bernama *stride* dan *padding*. *Stride* menentukan jumlah *pixels* yang bergeser secara horizontal dan vertikal. Semakin kecil *stride* yang digunakan, maka akan mendapatkan hasil yang lebih detail namun juga membutuhkan komputasi yang lebih baik. Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang baik (S).

Sedangkan *padding* adalah parameter yang menentukan jumlah *pixels* (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi output dari *convolutional layer* (*Feature Map*) agar tetap sama seperti dimensi input atau setidaknya tidak berkurang secara drastis, sehingga kita bisa menggunakan *convolutional layer* yang lebih dalam sehingga lebih banyak *features* yang berhasil diekstrak, dan meningkatkan performa dari model karena *filter* akan fokus pada informasi yang sebenarnya yaitu yang berada di antara *zero padding* tersebut.

Untuk menghitung dimensi dari *feature map* dapat menggunakan persamaan dibawah ini:

$$outuput = \frac{N-F+2P}{S} + 1 \qquad (1.1)$$

dimana:

- N = Panjang/Tinggi Input
- F = Panjang/Tinggi *Filter*
- P = Zero Padding
- S = Stride

Beberapa hal yang perlu diperhatikan pada *convolutional layer* diantaranya:

- Ukuran ketebalan dari sebuah *filter* selalu mengikuti ketebalan/volume dari gambar *input* yang digunakan.
- Tinggi (dan lebar) *filter* (F) pada umumnya berukuran ganjil. Secara intuisi, *filter* berukuran ganjil memberikan representasi yang lebih baik karena mencakup bagian kiri dan kanan yang "seimbang".
- Dalam sebuah convolutional *layer*, *filters* yang digunakan berukuran sama, untuk kemudahan proses komputasi.
- Jumlah *filter* (K) yang digunakan dalam sebuah *convolutional layer* adalah kelipatan 2 (*powers of* 2). Beberapa *library* memiliki *subroutine* khusus untuk komputasi kernel/dimensi berkelipatan 2 yang dapat meningkatkan efisiensi.

- Besaran *zero padding* (P) umumnya menyesuaikan agar ukuran spasial dari *output* yang dihasilkan tetap sama dengan ukuran spasial input. (P = F-1/2).

Secara keseluruhan, bila input sebuah *convolutional layer* adalah gambar dengan ukuran W1 x H1 x D1, *output* dari *layers* tersebut adalah sebuah "gambar" baru dengan ukuran W2 x H2 x D2, seperti pada persamaan di bawah ini:

$$\mathrm{W2} = \frac{w1 - F + 2P}{S} + 1 = H2 \qquad (1.2)$$

Keterangan:

- K adalah jumlah *filter* yang digunakan.
- F adalah ukuran spasial dari *filter* (lebar/tinggi).
- S adalah *stride*, atau besar pergeseran *filter* dalam konvolusi.
- P adalah *padding*, jumlah penambahan nol pada gambar.

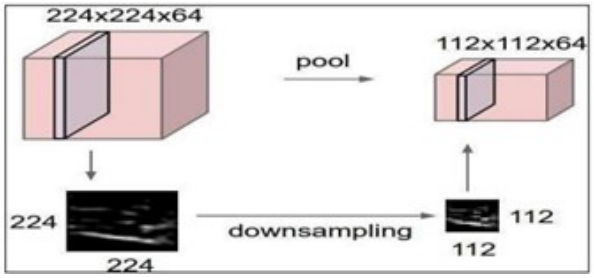
- Fungsi Aktivasi

Fungsi aktivasi berada pada tahap sebelum melakukan *pooling layer* dan setelah melakukan proses konvolusi. Pada tahap ini, nilai hasil konvolusi dikenakan fungsi aktivasi atau *activation function*. Terdapat beberapa fungsi aktivasi yang sering digunakan pada *convolutional network,* di antaranya t*anh()* atau *reLU.* Aktivasi *reLU* menjadi karena sifatnya yang lebih berfungsi dengan baik. Fungsi yang digunakan untuk aktivasi pada reLU, fungsi reLU adalah nilai *output* dari *neuron* bisa dinyatakan sebagai nol jika *input*nya adalah *negatif.* Jika nilai *input* dari fungsi aktivasi adalah *positif*, maka *output* dari *neuron* adalah nilai *input* aktivasi itu sendiri.

- *Pooling Layer*

Pooling Layer menerima *output* dari *convolutional layer*, pada *layer* ini dimensi data akan dikurangi (*downsampling*). Prinsipnya *pooling layer* terdiri dari *filter* dengan ukuran tertentu dan *stride*/langkah kemudian bergeser ke seluruh *area feature map*. Tujuan dari penggunaan *pooling layer* adalah untuk mengurangi dimensi dari *feature map* (*downsampling*),

sehingga mempercepat komputasi karena parameter yang harus di-*update* semakin sedikit dan mengatasi *overfitting* (*Pengenalan Deep Learning Part 7: Convolutional Neural Network (CNN) | by Samuel Sena |Medium*, n.d.). Gambar 2.4 di bawah ini merupakan gambaran dari *downsampling*.



Gambar 2.4 *Downsampling*

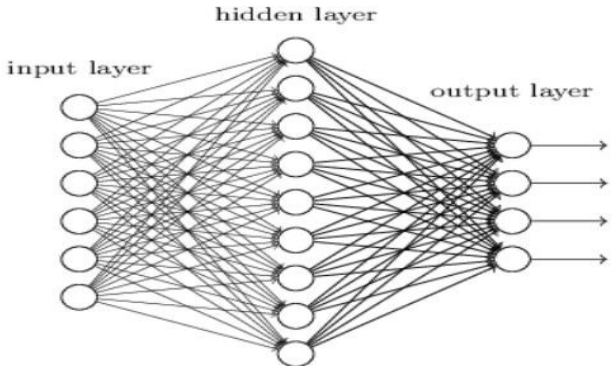
(Sumber: (*Pengenalan Deep Learning Part 7: Convolutional Neural Network (CNN) | by Samuel Sena |Medium*, n.d.))

2. *Fully Connected Layer*

Fully connected layer dalam Convolutional Neural Network (CNN) adalah lapisan akhir dari arsitektur CNN yang menghubungkan setiap neuron dalam lapisan sebelumnya secara penuh (*fully connected*) dengan setiap neuron dalam lapisan berikutnya. Lapisan ini juga dikenal sebagai lapisan Dense atau lapisan sepenuhnya terhubung.

Fully connected layer bertujuan untuk menggabungkan fitur-fitur tersebut dan melakukan klasifikasi atau regresi pada input. Setiap neuron dalam lapisan ini menerima input dari semua neuron di lapisan sebelumnya dan menghasilkan output yang kemudian akan diolah oleh fungsi aktivasi.

Fully Connected Layer mengambil *input* dari hasil *output pooling layer* yang berupa *feature map*. *Feature map* tersebut masih berbentuk *multi-dimensional array*. Oleh karena itu, lapisan ini akan melakukan *reshape feature map* dan menghasilkanvektor sebanyak n- dimensi, dimana n adalah jumlah kelas *output* yang harus dipilih program. Misalnya, jika lapisan terdiri dari 500 neuron, maka akan diterapkan fungsi aktivasi *softmax* yang mengembalikan daftar probabilitas terbesar untuk masing-masing 10 label kelas sebagai klasifikasi akhir dari jaringan (Feng & Law, 2021). Gambaran dari *fully connected layer* dapat dilihat pada gambar 2.5.



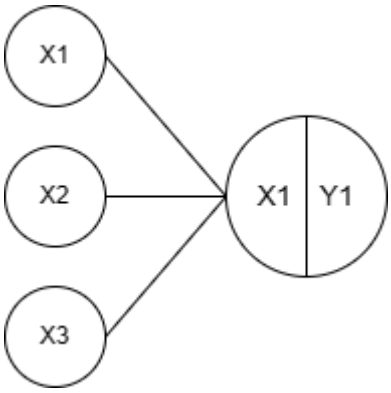
Gambar 2.5 *Fully Connected Layer*

(Sumber: (Feng & Law, 2021))

Adapun tahap dari *fully connected layer*:

- *Forward Propagation*

Forward Propagation merupakan salah satu proses pada jaringan syaraf tiruan. Dalam jaringan ini, informasi bergerak hanya satu arah, maju, dari *node input*, melalui *hidden layers* dan ke *node output*. Alur dari *backpropagation* dapat dilihat pada gambar 2.6.



Gambar 2.6 *Forward Propagation*

Untuk perhitungannya, nilai y_1 diperoleh dengan menghitung nilai z_1 terlebih dahulu dengan menggunakan perhitungan linier.

$$z_1 = w_{11}x_{11} + w_{21}x_2 + w_{23}x_3 + b_1 \quad (2.2)$$

Setelah diperoleh z_1, output prediksi y_1 diperoleh dengan menerapkan fungsi aktivasi terhadap z_1.

$$y_1 = \sigma(z_1) \quad (2.3)$$

Perhitungan untuk semua *y* secara umum bisa mengguanakn rumus:

$$y_j = \sigma\left(\sum_{i=1}^{\mathrm{N}} w_{ij}\, x_i + b_j\right) \tag{2.4}$$

Rumus di atas, sangat penting untuk dipahami, untuk penjelasan simbol – simbolnya:

- Simbol b_j menunjukkan nilai bias. Nilai bias ini mirip dengan nilai bobot hanya saja tidak dikalikan dengan input. Tujuannya agar garis persamaan bisa lebih kompleks (tidak selalu melewati titik origin).
- Semua nilai bobot *w* dan bias *b* awalnya diberikan nilai random, dan diperbarui nilainya dengan proses backprop untuk meningkatkan kualitas model. Jika diperhatikan kita menamai simbol w_{ij} berarti bobot yang menghubungkan neuron *input* nomor *i* ke neuron *output* nomor *j*.
- N menunjukkan banyak neuron di *layer* sebelah kiri (*layer input*).
- Simbol σ() (sigma) adalah simbol dari fungsi aktivasi. Artinya, setelah proses perkalian input *x* dan bobot *w* lalu dilakukan penjumlahan semua, langkah selanjutnya adalah mengenai hasil perhitungan tersebut dengan fungsi aktivasi.

- Error

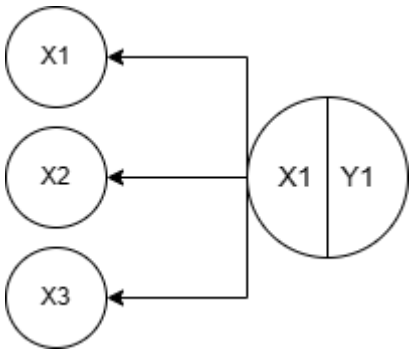
Dari perhitungan sebelumnya diperoleh nilai y_j sebagai nilai prediksi, atau nilai yang dihasilkan oleh Jaringan Syaraf tiruan. Seperti disebutkan sebelumnya setiap data yang masuk memiliki label kelas atau nilai *y* yang diharapkan. Untuk menghitung error. Salah satunya dengan menggunakan rumus *Mean Square Error* (MSE):

$$error = \frac{1}{n}\sum_{i=1}^{n}(target_i - prediksi_i)^2 \tag{2.5}$$

Rumus tersebut menghitung selisih nilai target dan prediksi, mengkuadratkannya, lalu merata-ratanya (dijumlah lalu dibagi *n*). Nilai *n* di sana adalah banyak datanya. Karena tujuan JST adalah untuk menghasikan nilai prediksi *y* yang semirip mungkin dengan *t*, maka dapat disebut juga tujuan dari JST adalah meminimalkan nilai *error*.

- Backpropagation

Setelah mendapatkan nilai error, JST nbisa diperbaiki dengan menggunakan backpropagation. Sebenarnya istilah memperbaiki JST ini juga disebut dengan Gradient Descent, di Indonesia lebih umum menyebutnya Backpropagation atau Backprop. Alur dari *backpropagation* dapat dilihat pada gambar 2.7.



Gambar 2.7 *Backpropagation*

Rumus utama untuk memperbaiki suatu bobot *w* berdasarkan error *E* adalah

$$w_{new} = w_{old} - \alpha \frac{\partial E}{\partial w} \tag{2.6}$$

Rumus ini juga berlaku untuk memperbaiki nilai bias:

$$b_{new} = b_{old} - \alpha \frac{\partial E}{\partial b} \tag{2.7}$$

Simbol α pada rumus di atas adalah learning rate, sebuah konstanta (biasanya antara 0-1) yang menentukan seberapa cepat proses pembelajaran model dilakukan. Simbol $\frac{\partial E}{\partial b}$ atau dibaca "turunan parsial E terhadap w" adalah proses mencari nilai turunan E terhadap variabel yang akan diperbarui.

- Mengupdate Bobot

Untuk memperbarui bobot, hitung semua perubahan pada bobot. Rumus perubahan bobot garis menuju ke unit keluaran sebagai berikut:

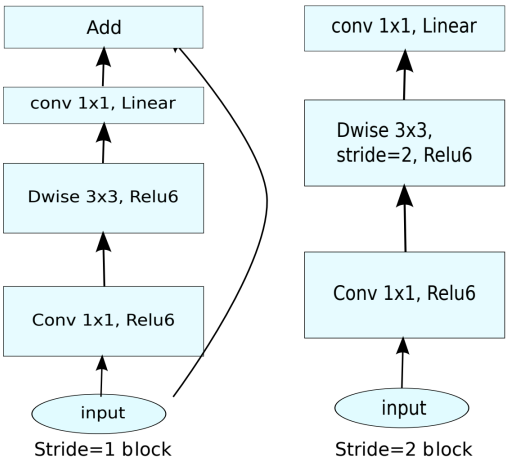
$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \ (k = 1,2,\dots,m; j = 0,1,\dots,p) \tag{2.8}$$

Perubahan bobot garis yang menuju ke unit tersembunyi :

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \ (j = 1,2,\dots,p; I = 0,1,\dots,n) \tag{2.9}$$

2.2.7 MobileNetV2

MobileNet, merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang dapat digunakan untuk mengatasi kebutuhan akan *computing resource* berlebih. Seperti namanya, para peneliti dari Google membuat arsitektur CNN yang dapat digunakan untuk ponsel atau perangkat dengan komputasi rendah. Jaringan pada arsitektur MobileNetV2 secara signifikan mengurangi jumlah operasi dan memori yang dibutuhkan sambil mempertahankan akurasi yang sama. Perbedaan MobileNetV2 dengan MobileNet versi sebelumnya terletak pada modul lapisan *inverterd residual* dengan *linear bottleneck* (Sandler dkk., 2019). Alur sistem dari blok konvolusi MobileNetV2 dapat dilihat pada gambar 2.8.



Gambar 2.8 Blok Konvolusi MobileNetV2

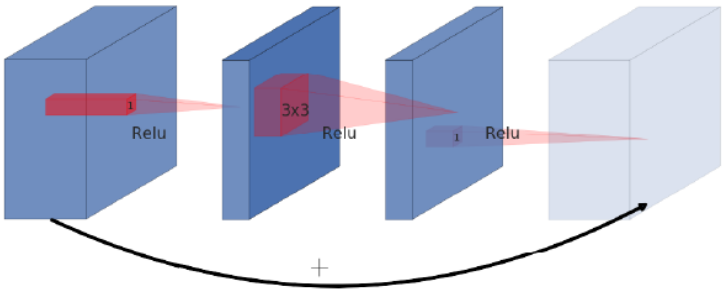
(Sumber: (Sandler dkk., 2019))

Modul ini mengambil sebagai masukan representasi terkompresi dimensi rendah yang pertama kali diperluas ke dimensi tinggi dan difilter dengan konvolusi mendalam yang ringan. Fitur kemudian diproyeksikan kembali ke representasi dimensi rendah dengan konvolusi linier. Secara keseluruhan, ini adalah perakitan blok konvolusi yang sangat sederhana dan umum.

- Inverted Residual

  Residual Block menghubungkan awal dan akhir blok konvolusi dengan koneksi lewati. Dengan menambahkan dua status ini, jaringan memiliki peluang untuk mengakses aktivasi sebelumnya yang tidak dimodifikasi di blok konvolusi. Pendekatan ini penting untuk membangun jaringan yang sangat

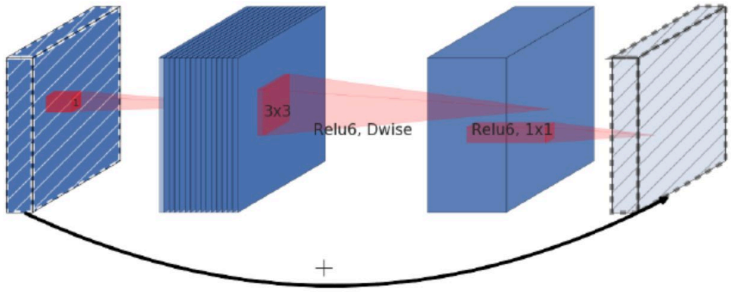
mendalam. Bagaimana alur dari *residual block* bekerja dapat dilihat pada gambar 2.9.



Gambar 2.9 *Residual Block*

(Sumber: (Sandler dkk., 2019))

Pada penelitian milik (Sandler dkk., 2019) menggunakan *inverted residual block* karena koneksi lewati ada di antara bagian jaringan yang sempit yang berlawanan dengan cara kerja koneksi sisa asli. Selain itu *inverted residual block* memiliki parameter yang jauh lebih sedikit dibandingkan *residual block*. Bagaimana alur dari *inverted residual block* bekerja dapat dilihat pada gambar 2.10.



Gambar 2.10 *Inverted Residual Block*

(Sumber: (Sandler dkk., 2019))

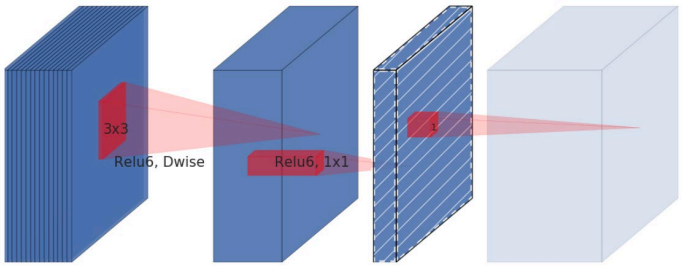
- Linear Bottlenecks

Pada penelitian (Sandler dkk., 2019) telah menyoroti dua properti yang menunjukkan persyaratan bahwa *manifold* yang menarik harus terletak pada subruang dimensi rendah dari ruang aktivasi dimensi tinggi:

- Jika *manifold of interest* tetap *non-zero volume* setelah transformasi ReLU, maka sesuai dengan transformasi linier.

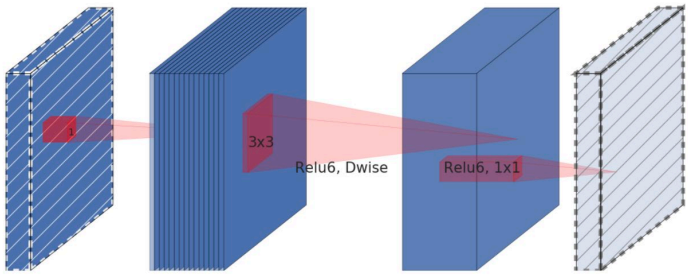
- ReLU mampu menyimpan informasi lengkap tentang manifold input, tetapi hanya jika manifold input terletak di subruang dimensi rendah dari ruang input

Kedua diatas memberikan petunjuk empiris untuk mengoptimalkan arsitektur saraf yang ada. dengan asumsi berbagai kepentingan berdimensi rendah, dapat ditangkap ini dengan memasukkan lapisan kemacetan linier ke dalam blok konvolusi. Gambar 2.11 merupakan gambaran ketika terjadi *bottleneck convolution*. Sedangkan Gambar 2.12 merupakan gambaran ketika terjadi *expansion convolution block*.



Gambar 2.11 *Bottleneck Convolution*

(Sumber: (Sandler dkk., 2019))



Gambar 2.12 *Expansion Convolution Block*

(Sumber: (Sandler dkk., 2019))

Evolusi blok konvolusi yang dapat dipisahkan. Tekstur menetas secara diagonal menunjukkan lapisan yang tidak mengandung non-linearitas. Lapisan terakhir (berwarna terang) menunjukkan awal dari blok berikutnya.

2.2.8 TensorFlow

TensorFlow adalah *library open-source* yang memungkinkan komputasi kinerja tinggi dan pembelajaran mesin (Ramadhan, 2022). TensorFlow digunakan

di banyak perusahaan mulai dari Nvidia hingga Intel, dan banyak lagi. TensorFlow beroperasi menggunakan array multidimensi yang disebut tensor. TensorFlow memungkinkan Python untuk menyelesaikan komputasi yang lebih rumit yang diperlukan saat bekerja dengan pembelajaran mesin. Format penyimpanan informasi ini digunakan untuk menyimpan informasi yang jauh lebih rumit dalam tensor daripada dalam array satu dimensi biasa. Sedangkan TensorFlow Lite digunakan untuk menerapkan model pembelajran di beberapa perangkat dan platform seperti seluler (iOS dan Android), desktop, dan perangkat edge lainnya.

Gambar 2.13 Logo Tensorflow

2.2.9 OpenMV Cam H7+

OpenMV Cam M7 Smart Vision Camera adalah papan mikrokontroler kecil berdaya rendah yang memungkinkan untuk diimplementasikan aplikasi dengan mudah menggunakan visi mesin di dunia nyata. OpenMV Cam bisa diprogram dalam skrip Python tingkat tinggi (milik MicroPython Operating System) alih-alih C/C++ (Ramadhan, 2022).

Gambar 2.14 OpenMV Cam H7 Plus

(Sumber: (Abdelkader dkk., 2017))

2.2.10 Confusion Matrix

Confusion Matrix adalah pengukuran performa untuk masalah klasifikasi. 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) (Beauxis-Aussalet, 2014). Istilah-istilah tersebut biasa dirangkum sebagai suatu matriks yang disebut *confusion matrix* sebagaimana ditunjukkan pada berikut ini:

- *True Positive* (TP)

Klasifikasi gambar ukuran bibit ikan lele yang ada di dalam model dan memang benar hasil klasifikasi menunjukan label ukuran bibit ikan lele yang ada di dalam model. Contoh dari TP dapat dilihat pada tabel 2.2.

Tabel 2.2 *True Positive* pada Confusion Matrix

		Aktual			
	Label	A	B	C	D
Prediksi	A	1	2	3	4
	B	4	3	2	1
	C	2	3	4	1
	D	3	4	1	2

- *True Negative* (TN)

Klasifikasi gambar ukuran bibit ikan lele yang tidak ada di dalam model dan memang benar label klasifikasi gambar ukuran bibit ikan lele yang tidak ada dalam model. Contoh dari TP dapat dilihat pada tabel 2.3.

Tabel 2.3 *True Negative* pada Confusion Matrix

		Aktual			
	Label	A	B	C	D
Prediksi	A	1	2	3	4
	B	4	3	2	1
	C	2	3	4	1
	D	3	4	1	2

- *False Positive* (FP)

Klasifikasi gambar ukuran bibit ikan lele yang ada di dalam model dan hasil klasifikasi menunjukan klasifikasi yang tidak sesuai dengan label ukuran bibit ikan lele. Contoh dari FP dapat dilihat pada tabel 2.4.

Tabel 2.4 *False Positive* pada Confusion Matrix

		Aktual			
	Label	A	B	C	D
Prediksi	A	1	2	3	4
	B	4	3	2	1
	C	2	3	4	1
	D	3	4	1	2

- *False Negative* (FN)

Klasifikasi gambar ukuran bibit ikan lele yang tidak ada di dalam model, tetapi hasil klasifikasi menunjukan bahwa klasifikasi tersebut menunjukkan label ukuran bibit ikan lele yang sesuai. Contoh dari FN dapat dilihat pada tabel 2.5.

Tabel 2.5 *False Negative* pada Confusion Matrix

		Aktual			
	Label	A	B	C	D
Prediksi	A	1	2	3	4
	B	4	3	2	1
	C	2	3	4	1
	D	3	4	1	2

Akurasi didefinisikan sebagai persentase dari data uji yang diklasifikasikan ke kelas yang benar. Akurasi dapat dinyatakan dalam persamaan (2.10):

$$akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.10)$$

Precision menggambarkan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Seperti pada persamaan (2.11):

$$\text{precision} = \frac{TP}{TP + FP} \quad (2.11)$$

Recall atau *sensitivity*, menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Persamaan *recall* dapat dilihat pada persamaan (2.12):

$$\text{recall} = \frac{TP}{TP + \text{FN}} \quad (2.12)$$

F-1 *Score* menggambarkan perbandingan rata-rata *precision* dan *recall* yang dibobotkan. *Accuracy* tepat digunakan sebagai acuan performansi algoritma

datasetkita memiliki jumlah data *False Negatif* dan *False Positif* yang sangat mendekati (*symmetric*). Namun, jika jumlahnya tidak mendekati maka sebaiknya menggunakan F1 *Score* sebagai acuan.

$$\text{F} - 1\ \text{score} = \frac{2\ \text{x recall x precision}}{\text{recall} + \text{precision}} \qquad (2.13)$$