BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian yang telah dilakukan sebelumnya berkaitan dengan judul proyek akhir ini, antara lain:

Penelitian yang dilakukan oleh Ramdhan dkk. (2024) mengembangkan sistem *smart trash bin* dengan memanfaatkan IoT dan SSD MobileNet untuk mempermudah pemilahan dan pengelolaan sampah di lingkungan kampus. Sistem ini menggunakan Raspberry Pi sebagai kontroler utama yang terintegrasi dengan kamera dan sensor ultrasonik untuk memproses gambar menggunakan algoritma Faster R-CNN dan SSD MobileNet, dengan bantuan TensorFlow untuk mendeteksi dan mengklasifikasikan jenis sampah. Berdasarkan klasifikasi tersebut, motor servo mengarahkan sampah untuk mengarahkan sampah ke tempat yang sesuai. Selain itu, sistem juga dilengkapi dengan sensor ultrasonik dan *bot* Telegram untuk memantau kapasitas sampah dan memberikan notifikasi kepada pengguna. Hasil penelitian tersebut menunjukkan bahwa sistem ini efektif dalam memisahkan sampah secara otomatis, meningkatkan efisiensi dan mempercepat pengelolaan sampah yang ramah lingkungan dan memberikan solusi bagi pengelolaan sampah di kampus.

Penelitian yang dilakukan oleh Sallang dkk. (2021) mengembangkan sistem pengelolaan sampah pintar berbasis CNN dan IoT untuk mempermudah klasifikasi dan pemantauan sampah pada lingkungan perkotaan. Sistem ini memanfaatkan model SSD MobileNetV2 Quantized yang diimplementasikan pada Raspberry Pi 4 yang dilatih untuk mengidentifikasi lima jenis sampah, yaitu kertas, kardus, kaca, logam, dan plastik. Kamera yang terhubung dengan Raspberry Pi menangkap gambar sampah, yang kemudian diklasifikasikan oleh model CNN, Sementara motor servo digunakan untuk mengendalikan papan plastik agar sampah masuk ke kategori yang sesuai. Selain itu sensor ultrasonik juga digunakan untuk memantau tingkat kepenuhan tempat sampah, dan modul GPS yang melacak lokasi secara *real-time*. Hasil penelitian menunjukkan akurasi dengan rata-rata

di atas 80% dan pemantauan jarak jauh yang dapat membantu mengurangi frekuensi pengumpulan sampah yang tidak perlu.

Penelitian yang dilakukan oleh Setiana (2024) mengembangkan sistem robot pemilah sampah berbasis Raspberry Pi menggunakan algoritma CNN dengan arsitektur MobileNet. Sistem tersebut dirancang untuk meningkatkan efisiensi pemilahan sampah di kampus, khususnya pada jenis sampah anorganik, organik, dan botol plastik. Model CNN MobileNet ini dipilih karena kemampuannya dalam mengklasifikasikan gambar secara akurat pada perangkat Raspberry Pi. Sistem ini juga dilengkapi dengan bot Telegram yang bertujuan untuk memberikan notifikasi kepada pengguna melalui aplikasi telegram ketika tempat sampah telah penuh. Hasil dari penelitian ini menunjukkan bahwa sistem robot pemilah sampah yang dirancang mampu mencapai akurasi hingga 90,53% dalam klasifikasi jenis sampah.

Penelitian yang dilakukan oleh Reza Fahcruroji dkk. (2024) mengimplementasikan algoritma CNN dengan arsitektur MobileNet untuk klasifikasi sampah dalam enam kategori sampah (plastik, metal, kertas, kardus, kaca, dan organik) pada Bank sampah menggunakan *dataset* dari Kaggle. Penelitian ini menggunakan metode *preprocessing* untuk mengubah ukuran gambar menjadi 224x224 piksel dan melakukan augmentasi data. Model dilatih dengan menggunakan 15 *epoch* menggunakan Google Colab untuk proses komputasi. Hasilnya menunjukkan akurasi yang bervariasi untuk setiap kategori, yaitu 96% untuk metal, 92% untuk kertas dan organik, 80% untuk kardus, 76% untuk kaca, dan 72% untuk plastik. Hasil dari penelitian ini menunjukkan bahwa CNN MobileNet memiliki akurasi yang baik dalam mengklasifikasikan sampah sehingga dapat memudahkan dalam pendataan dan pengelolaan Bank sampah lebih baik lagi.

Penelitian yang dilakukan oleh Muchtar dkk. (2022) mengembangkan *prototype* pemilahan sampah pintar berbasis *deep learning* dengan memanfaatkan CNN dan arsitektur SSD MobileNet untuk klasifikasi sampah menjadi kategori organik dan anorganik. Sistem ini menggunakan Raspberry Pi sebagai mikrokontroler utama yang terintegrasi dengan kamera Raspberry Pi dan Movidius Neural Compute Stick untuk mempercepat proses komputasi. *Dataset* yang digunakan terdiri atas lima jenis sampah utama yaitu plastik, kertas, kaca, metal, dan

kardus dengan total 2.390 gambar yang diambil dari repositori publik. *Prototype* ini diuji dengan mendeteksi dan mengklasifikasikan sampah secara *real-time*. Jika sampah berhasil diidentifikasi, tempat sampah yang sesuai akan terbuka secara otomatis, dan informasi gambar yang terdeteksi beserta nilai akurasinya akan ditampilkan pada *website* untuk pemantauan. Hasil dari penelitian ini menunjukkan bahwa sistem mampu mencapai akurasi rata-rata mencapai 98,84% sehingga berhasil menciptakan solusi untuk pemilahan sampah yang efektif, efisien, dan biaya yang rendah untuk mendukung pengelolaan sampah.

Penelitian yang akan dilakukan oleh penulis dalam proyek akhir ini akan mengembangkan sebuah sistem klasifikasi sampah berbasis *Convolutional Neural Network* (CNN). Sistem ini akan memanfaatkan perangkat keras berbasis Raspberry Pi yang dilengkapi dengan kamera dan sensor ultrasonik untuk mendeteksi dan mengklasifikasikan sampah menjadi kategori organik dan anorganik. Model CNN yang digunakan dalam penelitian ini adalah MobileNetV2, yang dipilih karena efisiensinya dalam pengolahan gambar dengan sumber daya terbatas, sehingga cocok untuk dijalankan pada Raspberry Pi. Model MobileNetV2 akan dilatih menggunakan data gambar yang telah dikumpulkan. Setelah proses klasifikasi selesai, sistem akan mengontrol motor servo untuk memutar tempat peletakan sampah ke arah tong yang sesuai dengan kategori sampah yang terdeteksi.

Tabel 2.1 Penelitian Terdahulu

No.	Nama Peneliti	Judul Penelitian	Algoritma	Hasil
1	(Ramdhan dkk., 2024)	Pemanfaatan Teknologi IoT pada Smart Trash Bin untuk Pemilahan dan Monitoring Sampah	Faster R-CNN dan MobileNet	Sistem dapat memisahkan sampah secara otomatis dengan akurasi yang tinggi dan mengirim notifikasi saat tempat sampah penuh.

No.	Nama Peneliti	Judul Penelitian	Algoritma	Hasil
2	(Sallang dkk., 2021)	A CNN-Based Smart Waste Management System using Tensorflow Lite and LoRa-GPS Shield in Internet of Things Environment	SSD MobileNetV2 Quantized	Sistem dapat mendeteksi dan mengklasifikasi lima jenis sampah (kertas, kardus, kaca, logam, plastik) serta memantau tingkat kepenuhan sampah secara *real-time*
3	(Setiana, 2024)	Robot Pemilahan Sampah Berbasis Raspberry Pi Menggunakan Algoritma Convolutional Neural Network dengan MobileNet	MobileNet	Sistem dapat mendeteksi dan memisahkan sampah organik, anorganik, dan botol plastik dengan akurasi model 90,53%, serta memberikan notifikasi saat tempat sampah penuh
4	(Reza Fahcruroji dkk., 2024)	Implementasi Algoritma CNN MobileNet untuk Klasifikasi Gambar Sampah di Bank Sampah	MobileNetV2	Model dilatih dengan 15 *epoch* menggunakan *preprocessing* dan augmentasi data. Akurasi mencapai 96% untuk metal, 92% untuk kertas dan

No.	Nama Peneliti	Judul Penelitian	Algoritma	Hasil
				organik, 80% kardus, 76% kaca, dan 72% plastik.
5	(Muchtar dkk., 2022)	Rancang Bangun Purwarupa Pemilahan Sampah Pintar Berbasis *Deep Learning*	SSD MobileNet, Intel Movidius NCS	Sistem dapat mendeteksi lima jenis sampah dengan akurasi rata-rata 98,84% dan menampilkan hasil deteksi pada situs web yang terintegrasi untuk pemantauan *real-time*.

2.2 Landasan Teori

2.2.1 Sampah

Menurut World Health Organization (WHO), sampah adalah sesuatu yang tidak digunakan, tidak dipakai, tidak disenangi, atau sesuatu yang dibuang yang berasal dari kegiatan manusia dan tidak terjadi dengan sendirinya. Menurut Undang-Undang Nomor 8 Tahun 2008 Tentang Pengelolaan Sampah, sampah didefinisikan sebagai sisa dari aktivitas manusia sehari-hari atau proses alam yang memiliki bentuk padat atau semi-padat, terdiri dari zat organik maupun anorganik, serta bersifat dapat terurai maupun tidak. Sampah dianggap tidak memiliki nilai guna sehingga dibuang ke lingkungan.

2.2.1.1 Sampah Organik

Sampah organik adalah jenis sampah yang berasal dari sisa-sisa makhluk hidup, seperti hewan, tanaman, dan manusia. Sampah ini dapat terurai secara alami dan termasuk sisa makanan, potongan buah dan sayur, dedaunan, serta kotoran hewan (Defitri, 2023). Sampah organik

dapat dimanfaatkan menjadi kompos atau pakan ternak. Menurut data dari Sistem Informasi Pengelolaan Sampah Nasional (SIPSN), sampah organik mendominasi komposisi sampah di Indonesia, dengan total yang mencapai lebih dari setengah total timbunan sampah nasional. Pada tahun 2023, sisa makanan menyumbang sekitar 39,65% dari total sampah organik, yang terdiri atas sisa buah-buahan, sayuran, dan makanan yang tidak habis. Selain itu, jenis sampah organik lainnya berasal dari kayu, ranting, dan daun, yang mencapai 12,09%. Secara keseluruhan, sampah organik berkontribusi sebesar 51,74% terhadap total sampah nasional (SIPSN - Sistem Informasi Pengelolaan Sampah Nasional, 2024).

Gambar 2.1 Sampah Organik (Sumber: waste4change.com)

2.2.1.2 Sampah Anorganik

Sampah Anorganik adalah jenis sampah yang tidak dapa terurai secara alami karena terbuat dari material yang tidak berasal dari alam, melainkan hasil olahan dari bahan sintetik (Defitri, 2023). Contoh sampah anorganik termasuk kantong plastik, kaleng, botol kaca, karton, dan sebagainya. Menurut data dari Sistem Informasi Pengelolaan Sampah Nasional (SIPSN), sampah organik mendominasi komposisi sampah di Indonesia dengan total mencapai lebih dari setengah dari total timbulan sampah nasional. Pada tahun 2023, plastik menjadi sampah anorganik terbesar dengan kontribusi sebesar 19,21% dari total sampah, meliputi beragam jenis plastik seperti botol, kantong, dan kemasan. Sampah kertas dan karton mengikuti dengan persentase 10,83%, kemudian disusul oleh logam sebesar 3,24%, termasuk logam aluminium dan besi. Selain itu, sampah kaca seperti botol dan produk kaca lainnya mencapai 2,46% (SIPSN - Sistem Informasi Pengelolaan Sampah Nasional, 2024).

Gambar 2.2 Sampah Anorganik (Sumber: waste4change.com)

2.2.2 *Deep Learning*

Deep Learning (DL) adalah salah satu cabang dari *Machine Learning* (ML) yang terinspirasi oleh pola pemrosesan informasi pada otak manusia. *Deep learning* menggunakan jaringan saraf tiruan (*Artificial Neural Networks*, ANNs) dengan banyak lapisan untuk mengekstrak pola yang kompleks dari data dalam jumlah besar. *Deep learning* memungkinkan pembelajaran representasi data secara otomatis melalui proses transformasi yang dilakukan oleh berbagai algoritma. Hal ini berbeda dengan pendekatan tradisional dalam *machine learning* yang sering kali memerlukan rekayasa fitur manual untuk meningkatkan performa model (Alzubaidi dkk., 2021).

Deep learning mampu menangani berbagai jenis data seperti gambar, teks, dan suara. Salah satu keunggulan utama dari *deep learning* ini adalah kemampuannya dalam mengekstraksi fitur secara otomatis melalui proses pembelajaran secara berlapis. Dalam arsitektur ini, lapisan awal biasanya mempelajari fitur dasar, seperti tepi atau pola pada gambar, sedangkan lapisan yang lebih dalam menangkap representasi fitur yang lebih abstrak dan kompleks. Keunggulan ini membuat DL sangat efektif dalam berbagai aplikasi seperti *image classification, speech recognition*, dan *natural language processing* (NLP) (Alzubaidi dkk., 2021).

Deep learning memiliki tiga kategori utama, yaitu *Supervised Learning*, *Semi-supervised Learning*, dan *Unsupervised learning* (Alzubaidi dkk., 2021).

a) *Supervised learning*

Supervised learning adalah teknik pembelajaran yang bergantung pada data berlabel, di mana pasangan *input* dan *output* digunakan untuk melatih model. Model ini mempelajari hubungan antara *input* dan *output* melalui proses optimasi fungsi loss untuk meminimalkan kesalahan prediksi. Pendekatan ini sering digunakan dengan algoritma seperti *Convolutional Neural Networks* (CNNs), *Recurrent Neural Networks* (RNNs), dan *Deep Neural Networks* (DNNs).

b) *Semi-supervised learning*

Semi-supervised learning adalah pendekatan yang menggunakan kombinasi data berlabel dan tidak berlabel untuk melatih model. Teknik ini bertujuan untuk mengurangi kebutuhan akan data berlabel yang besar dengan memanfaatkan data tidak berlabel untuk meningkatkan akurasi model. Contoh populer termasuk pengklasifikasi dokumen teks di mana data berlabel sulit diperoleh. Algoritma seperti *Generative Adversarial Networks* (GANs) dan beberapa bentuk RNN sering digunakan dalam semi-supervised learning.

c) *Unsupervised learning*

Unsupervised learning adalah proses pembelajaran tanpa data berlabel. Model dirancang untuk mengenali struktur atau hubungan tersembunyi dalam data input. Teknik seperti *generative networks*, *clustering*, dan *dimensionality reduction* sering digunakan dalam kategori ini. *Restricted Boltzmann Machines* (RBMs), *Autoencoders*, dan *GANs* adalah beberapa algoritma populer dalam unsupervised learning.

Namun, dalam banyak kasus, model *deep learning* memerlukan jumlah data yang besar. Keterbatasan *dataset* dapat menyebabkan *overfitting*, di mana model terlalu menyesuaikan diri dengan data pelatihan dan gagal mengenali pola baru saat diuji dengan data yang belum pernah dilihat sebelumnya. Untuk mengatasi permasalahan ini, teknik augmentasi data digunakan untuk meningkatkan variasi dalam *dataset*. Augmentasi data (*Data Augmentation*) adalah teknik yang digunakan untuk meningkatkan variasi *dataset* pelatihan dengan melakukan transformasi pada data yang sudah ada. Teknik ini digunakan untuk mengatasi masalah *overfitting* serta meningkatkan generalisasi model, terutama ketika jumlah data yang tersedia terbatas. Pada

pengolahan citra digital, augmentasi data mencakup berbagai transformasi seperti:

- Rotasi (*Rotation*) – Memutar gambar dalam rentang sudut tertentu.
- Translasi (*Translation*) – Menggeser gambar secara horizontal atau vertikal.
- *Zooming* – Memperbesar atau memperkecil gambar.
- *Flipping* (*Mirror Effect*) – Membalik gambar secara horizontal atau vertikal.
- *Brightness Adjustment* – Mengubah kecerahan gambar.
- *Noise Injection* – Menambahkan *noise* ke dalam gambar untuk membuat model lebih tahan terhadap variasi *input* di dunia nyata.

Dengan menerapkan augmentasi data, model dapat lebih baik mengenali variasi gambar dalam dunia nyata dan meningkatkan akurasi prediksi pada data yang belum pernah dilihat sebelumnya.

Dalam proses pelatihan model deep learning, terdapat beberapa konsep penting yang memengaruhi performa model, yaitu:

1) Epoch

Epoch adalah satu siklus lengkap di mana seluruh *dataset* digunakan untuk melatih model. Dalam satu *epoch*, model melihat setiap sampel data satu kali. Jumlah *epoch* yang tepat bergantung pada kompleksitas model dan *dataset*. Terlalu sedikit *epoch* dapat menyebabkan model belum cukup belajar (*underfitting*), sedangkan terlalu banyak *epoch* dapat menyebabkan *overfitting*, di mana model terlalu spesifik terhadap data pelatihan dan tidak generalisasi dengan baik pada data baru.

2) *Learning Rate*

Learning rate adalah *hyperparameter* yang mengontrol seberapa besar perubahan bobot model selama proses pelatihan. Nilai *learning rate* yang terlalu kecil dapat membuat proses pelatihan berlangsung lama dan mungkin terjebak pada local minimum. Sebaliknya, nilai *learning rate* yang terlalu besar dapat menyebabkan model tidak stabil atau bahkan gagal mencapai konvergensi.

3) *Batch Size*

Batch size adalah jumlah sampel data yang diproses oleh model sebelum bobotnya diperbarui. *Batch size* yang lebih kecil membutuhkan lebih banyak iterasi untuk menyelesaikan satu *epoch*, tetapi dapat mengurangi penggunaan memori. *Batch size* yang lebih besar dapat mempercepat pelatihan tetapi membutuhkan lebih banyak memori.

4) *Loss Function*

Loss Function adalah metrik yang mengukur seberapa baik model melakukan prediksi terhadap data pelatihan. Contoh *loss function* yang umum digunakan dalam *deep learning* adalah *Mean Squared Error* (MSE) untuk regresi dan *Cross-Entropy Loss* untuk klasifikasi. Model akan mencoba meminimalkan nilai *loss function* melalui optimasi.

5) *Optimizer*

Optimizer adalah algoritma yang memperbarui bobot model berdasarkan nilai *loss function* agar model dapat belajar secara optimal. Salah satu yang sering digunakan adalah *Stochastic Gradient Descent* (SGD), yang memperbarui bobot berdasarkan sampel acak untuk menghindari jebakan *local minima*. Selain itu, *Adam* (*Adaptive Moment Estimation*) menggabungkan keunggulan SGD dengan momentum dan penyesuaian adaptif terhadap *learning rate*, sehingga cocok untuk berbagai arsitektur *deep learning*. Sementara itu, *RMSprop* (*Root Mean Square Propagation*) menyesuaikan *learning rate* berdasarkan rata-rata pergerakan kuadrat gradien, membuatnya efektif dalam data sekuensial seperti *Recurrent Neural Networks* (RNNs).

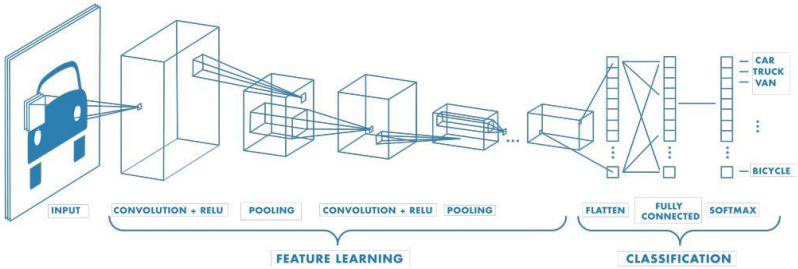
6) *Activation Function*

Fungsi aktivasi (*activation function*) adalah komponen penting dalam jaringan saraf tiruan yang memperkenalkan non-linearitas, memungkinkan model mempelajari pola kompleks. *ReLU* (*Rectified Linear Unit*) sering digunakan dalam lapisan tersembunyi karena meningkatkan efisiensi pelatihan dengan mengatasi *vanishing gradient*. *Sigmoid* cocok untuk klasifikasi biner, tetapi rentan terhadap *vanishing gradient*, sementara *tanh* memiliki rentang -1 hingga 1, yang dapat mempercepat

pembelajaran. Untuk klasifikasi multi-kelas, *softmax* digunakan karena mengubah *output* menjadi distribusi probabilitas.

2.2.3 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan algoritma *deep learning* yang dirancang untuk mengolah data berbentuk dua dimensi. CNN sering digunakan untuk mempelajari dan mendeteksi fitur dalam gambar (MatWorks, 2024). Berbeda dengan *neural network* pada umumnya yang menggunakan data *array* satu dimensi sebagai *input*, CNN menggunakan data dua dimensi sebagai *input*. CNN terdiri banyak neuron yang memiliki *weight*, *bias*, dan *activation function*. Gambar 2.3 menunjukkan tahapan algoritma CNN dalam melakukan proses *input* gambar.



Gambar 2.3 Tahapan proses CNN (MatWorks, 2024)

Pada Gambar 2.3 kita bisa melihat pada tahapan proses CNN ini, terdapat dua bagian utama yaitu, *feature learning* dan *classification*.

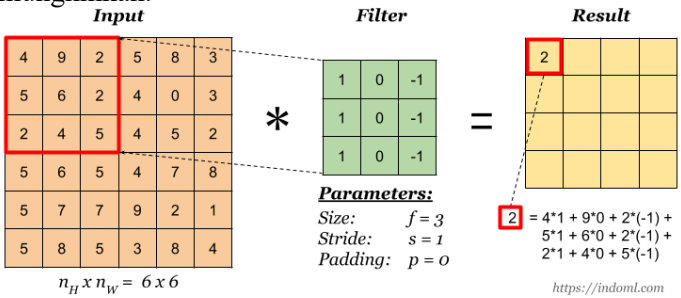
2.2.3.1 Feature Learning

Feature Learning merupakan bagian dari CNN yang bertujuan untuk mengekstraksi fitur-fitur penting dari data *input* (misalnya gambar). Bagian ini terdiri dari beberapa lapisan yang berperan dalam mengenali pola-pola lokal dalam gambar secara bertahap, dari pola sederhana hingga yang lebih kompleks. Berikut adalah penjelasan dari setiap *layer* yang terdapat pada *feature learning* yaitu (Alzubaidi dkk., 2021):

1) *Convolutional Layer*

Convolutional layer merupakan komponen utama dalam arsitektur CNN, yang terdiri dari sekumpulan filter konvolusi (kernel). Gambar *input*, yang berupa matriks N-dimensi, dikonvolusikan dengan kernel untuk menghasilkan peta fitur keluaran. Operasi Konvolusi dimulai dengan gambar masukan

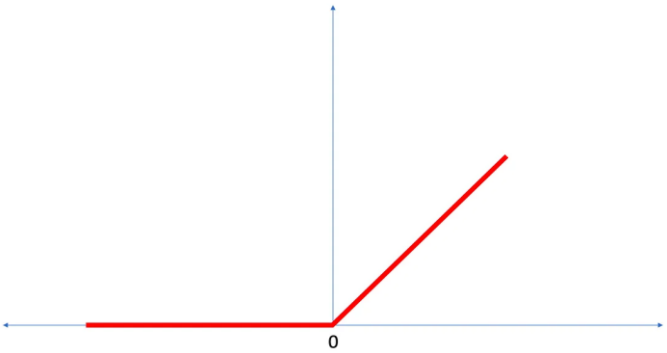
yang dapat berupa gambar *grayscale* (saluran tunggal) atau RGB (tiga saluran). Misalnya, pada gambar 6x6 dengan kernel 3x3, kernel ini digeser melintasi gambar secara horizontal dan vertikal. Pada setiap posisi, dilakukan perkalian titik antara kernel dan bagian gambar yang sesuai, dan hasilnya dijumlahkan untuk menghasilkan nilai skalar yang menjadi entri peta fitur. Proses ini diulang sampai tidak ada pergeseran lebih lanjut yang memungkinkan.



Gambar 2.4 Operasi 'dot' antara *input* dengan filter (Prijono, 2018)

2) *Rectified Linear Unit* (ReLu) *activation*

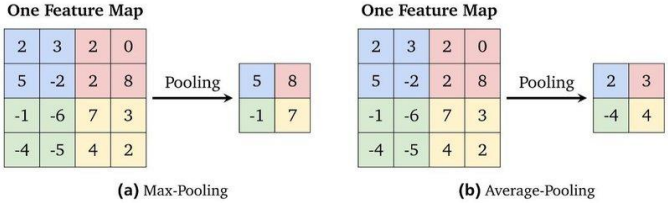
ReLu (*Rectified Linear Unit*) adalah fungsi aktivasi yang banyak digunakan dalam jaringan saraf tiruan (*neural network*), terutama dalam konteks *deep learning*. Fungsi ini bertujuan untuk memperkenalkan sifat non-linear pada model, yang penting agar jaringan saraf bisa memecahkan masalah yang lebih kompleks, seperti pengenalan gambar atau pemrosesan bahasa alami. *Ouput* dari fungsi aktivasi dinyatakan sebagai 0 (nol) jika *input-nya* negatif. Namun jika *input-nya* positif, maka *output-nya* sama dengan nilai *input* aktivasi itu sendiri.



Gambar 2.5 Grafik ReLu (Asyrofi, 2020)

3) Pooling Layer

Pooling Layer adalah salah satu komponen penting dalam arsitektur *Convolutional Neural Network* (CNN) yang digunakan untuk mengurangi dimensi dari peta fitur (*feature map*) dan mengurangi jumlah parameter yang perlu dipelajari, sambil mempertahankan informasi penting. Proses ini dikenal dengan istilah *down sampling*. terdapat dua jenis *pooling* yang umum digunakan, yaitu *max pooling* dan *average pooling*.



Gambar 2.6 Perbedaan *Max-Pooling* dan *Average-Pooling* (Edbert, 2021)

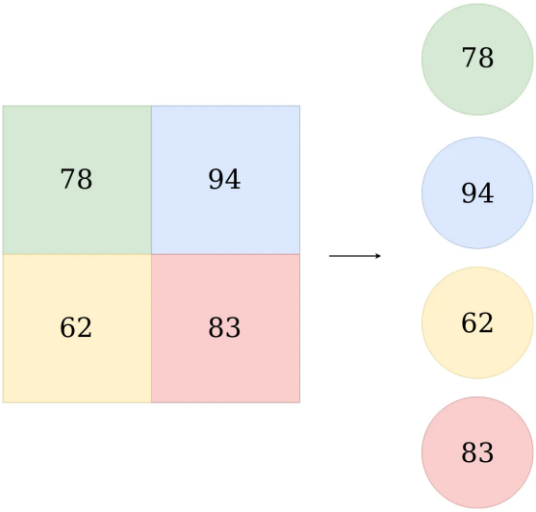
2.2.3.2 Classification

Classification berfungsi untuk mengklasifikasikan setiap neuron yang telah diekstraksi selama proses *feature learning*. Bagian ini terdiri dari beberapa lapisan yang saling terhubung satu sama lain. Berikut adalah penjelasan mengenai fungsi-fungsi yang ada pada bagian klasifikasi.

1) Flattening

Flattening adalah proses mengubah data berdimensi lebih tinggi (misalnya, matriks) menjadi vektor satu dimensi. Ini

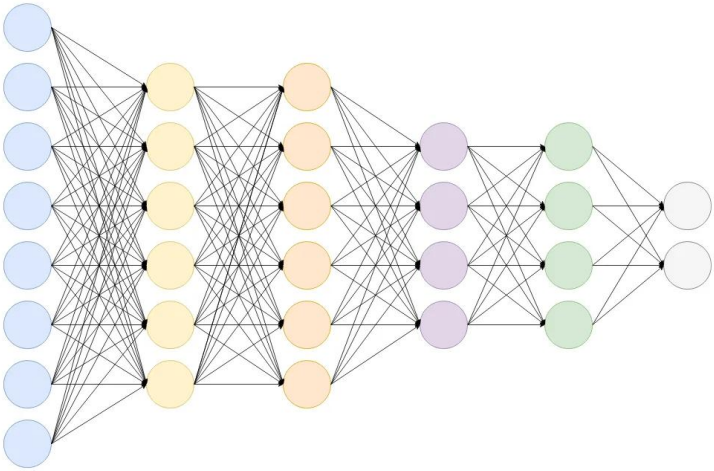
dilakukan untuk mempersiapkan data agar bisa diproses lebih lanjut oleh lapisan-lapisan selanjutnya dalam jaringan syaraf (*neural network*). Setelah fitur diekstraksi melalui lapisan konvolusi dan *pooling*, data biasanya berada dalam bentuk matriks atau *tensor* dan *flattening* akan meratakan matriks tersebut menjadi satu baris vektor untuk diproses di lapisan berikutnya.



Gambar 2.7 *Flatenning* (Arc, 2018)

2) *Fully Connected Layer*

Fully connected layer adalah bagian dari *feedforward neural networks* yang mencakup *hidden layer*, *activation function*, *output layer*, dan *loss function*. Lapisan ini sering diterapkan dalam *multi-layer perceptron* (MLP) dengan tujuan untuk mentransformasi dimensi data agar dapat diklasifikasikan secara linear. *Input* pada *fully connected layer* berasal dari data yang dihasilkan melalui *feature learning*. Data ini, yang sudah berbentuk vektor setelah proses *flattening*, kemudian diproses lebih lanjut dalam lapisan ini dan setiap neuron dihubungkan ke neuron-neuron di lapisan berikutnya, sehingga memungkinkan informasi untuk dipertimbangkan secara menyeluruh dan menghasilkan *output* yang lebih akurat.



Gambar 2.8 *Fully Connected Layer* (Arc, 2018)

3) SoftMax

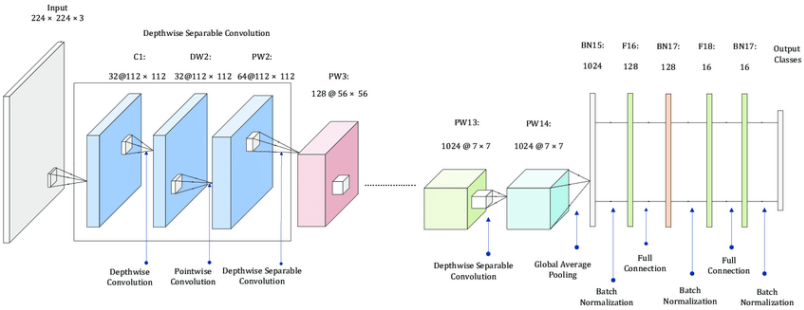
SoftMax adalah fungsi aktivasi yang sering digunakan pada lapisan terakhir dari *neural network* untuk klasifikasi multikelas Fungsi ini mengubah *output* dari jaringan menjadi probabilitas, yang dapat digunakan untuk menentukan kelas yang paling mungkin. Fungsi *SoftMax* mengonversi nilai-nilai logit yang tidak terbatas menjadi nilai probabilitas yang berada di antara 0 dan 1, dan jumlah total probabilitas untuk semua kelas selalu sama dengan 1.

2.2.4 MobileNetV2

MobileNetV2 merupakan arsitektur *Convolutional Neural Network* (CNN) yang dirancang khusus untuk aplikasi *mobile* dan sistem dengan sumber daya yang terbatas. MobileNetV2 merupakan pengembangan dari MobileNetV1, yang mengutamakan efisiensi dalam hal penggunaan sumber daya komputasi dan memori tanpa mengorbankan akurasi. MobileNetV2 memperkenalkan dua fitur baru yaitu *inverted residuals* dan *linear bottlenecks*. *Inverted Residuals* adalah desain blok residu terbalik yang menggunakan koneksi shortcut langsung antara lapisan *bottleneck* yang tipis, memungkinkan propagasi gradien yang lebih efisien sambil mempertahankan informasi penting pada representasi fitur. Sementara itu, *linear bottlenecks* menghilangkan fungsi non-linearitas pada lapisan *bottleneck* untuk mengurangi hilangnya

informasi penting di sub ruang berdimensi rendah, sehingga memperkuat kapasitas representasi model. (Sandler dkk., 2018).

Arsitektur MobileNetV2 juga mengadopsi *depthwise separable convolutions*, sebuah teknik yang membagi operasi konvolusi menjadi dua tahap: *depthwise convolution* untuk menyaring fitur pada setiap kanal secara independen, dan *pointwise convolution* untuk menggabungkan informasi antar kanal. Pendekatan ini secara signifikan mengurangi kebutuhan komputasi hingga 8-9 kali dibandingkan dengan konvolusi standar, tanpa mengorbankan akurasi. MobileNetV2 juga memanfaatkan ekspansi lapisan sementara untuk memungkinkan jaringan mengekstraksi fitur yang kompleks sebelum memproyeksikan kembali fitur-fitur tersebut ke representasi berdimensi rendah.



Gambar 2.9 Arsitektur MobileNetV2

(Source: https://pemrogramanmatlab.com/2023/07/23/jenis-jenis-arsitektur-convolutional-neural-network-cnn-untuk-image-recognition-dan-computer-vision/)

Gambar 2.9 menunjukkan arsitektur MobileNetV2 yang memanfaatkan *depthwise separable convolution*. Proses dimulai dengan memproses gambar *input* berdimensi 224×224×3 melalui lapisan *convolutional* pertama untuk menghasilkan representasi fitur awal. Operasi *depthwise separable convolution* kemudian diterapkan secara bertahap untuk meningkatkan jumlah kanal (misalnya dari 32 menjadi 640) dan mengurangi resolusi spasial (contohnya dari 112×112 menjadi 56×56) menggunakan operasi dengan *stride*. Pada tahap akhir, fitur dengan dimensi 7×7 dipadatkan menggunakan *Global Average Pooling* untuk menghasilkan representasi berdimensi rendah. Representasi ini diproses lebih lanjut melalui beberapa lapisan *fully connected* (FC) yang

dilengkapi *Batch Normalization* untuk meningkatkan stabilitas pelatihan dan mempercepat konvergensi. Akhirnya, lapisan *output* menghasilkan probabilitas untuk setiap kelas target sesuai dengan tugas klasifikasi.

2.2.5 Python

Python merupakan bahasa pemrograman tingkat tinggi yang dirancang untuk kemudahan penggunaan dan keterbacaan kode. Diciptakan oleh Guido van Rossum pada akhir 1980-an. Salah satu dari keunggulan python adalah sintaknya yang sederhana dan mirip dengan bahasa inggris, sehingga membuatnya mudah dipahami oleh pemula. Python memiliki komunitas besar yang secara aktif mengembangkan pustaka-pustaka tambahan yang memudahkan berbagai macam pengembangan aplikasi. Dengan lebih dari 125.000 pustaka yang tersedia, Python dapat digunakan untuk berbagai keperluan, seperti analisis data, *machine learning*, dan sebagainya. *Library* seperti Pandas, NumPy, dan Matplotlib menjadikan Python sebagai pilihan utama di bidang *data science* dan *machine learning* karena memungkinkan pemrogram untuk melakukan analisis data, visualisasi, serta membangun model algoritma *machine learning* dengan lebih efisien (Dicoding, 2023).

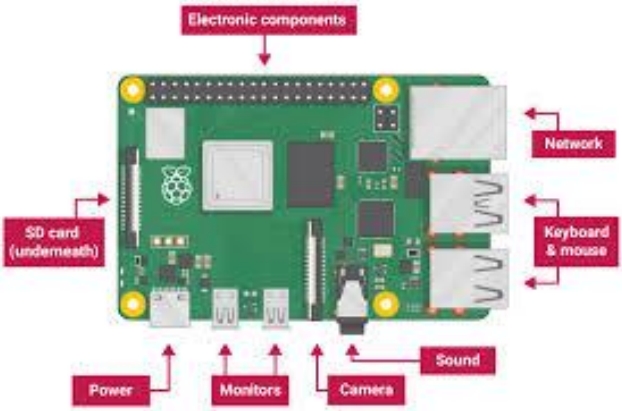
2.2.6 TensorFlow

TensorFlow adalah sebuah *framework open-source* yang digunakan untuk membangun dan melatih model *machine learning* dan *deep learning*. Dikembangkan oleh Google Brain pada tahun 2015, TensorFlow memungkinkan *developer* untuk membuat, melatih, dan menerapkan model-model *machine learning* yang kompleks dalam berbagai aplikasi. Selain itu, TensorFlow menyediakan berbagai alat dan *library* tambahan yang memungkinkan pengguna untuk melakukan berbagai tugas, mulai dari pelatihan model hingga penyebaran aplikasi *machine learning* ke berbagai perangkat, baik di *cloud*, perangkat *mobile*, maupun di perangkat keras khusus seperti TPU (*Tensor Processing Unit*) untuk komputasi yang lebih cepat. TensorFlow terus berkembang menjadi salah satu *framework* paling populer dalam dunia *machine learning*, memberikan solusi untuk penelitian, pengembangan produk, dan aplikasi komersial di berbagai sektor industri (TensorFlow, 2024).

2.2.7 Raspberry Pi

Raspberry Pi adalah sebuah komputer mini berukuran kartu kredit yang dirancang untuk memberikan akses komputasi dengan biaya

rendah dan mampu menjalankan berbagai aplikasi berbasis komputer. Raspberry Pi pertama kali dikembangkan oleh Raspberry Pi Foundation di Inggris pada tahun 2012 dengan tujuan untuk memfasilitasi pembelajaran komputer dan pemrograman, terutama di negara-negara berkembang. Secara umum, Raspberry Pi berfungsi sebagai komputer lengkap meskipun ukurannya sangat kecil. Raspberry Pi terdiri dari *motherboard* tunggal yang dilengkapi dengan berbagai *port* seperti HDMI, USB, dan GPIO (*General Purpose Input/Output*), yang memungkinkan pengguna untuk menghubungkan perangkat keras eksternal seperti monitor, *keyboard*, *mouse*, serta sensor dan perangkat lain untuk proyek elektronik. Raspberry Pi memanfaatkan kartu SD sebagai media penyimpanan utama, di mana sistem operasi berfungsi untuk menjalankan perangkat ini (BasuMallick, 2022).



Gambar 2.10 Raspberry Pi
(Sumber: https://bit.ly/3EBv46a)

2.2.8 Fritzing

Fritzing adalah perangkat lunak open-source yang digunakan untuk merancang, mendokumentasikan, dan membuat prototipe sirkuit elektronik secara visual. Perangkat lunak ini memiliki tiga tampilan utama, yaitu Breadboard View untuk menyusun komponen secara intuitif, Schematic View untuk menganalisis hubungan antar komponen, dan PCB View untuk merancang jalur sirkuit sebelum dicetak. Fritzing mudah digunakan, kompatibel dengan berbagai perangkat open-source seperti Arduino dan Raspberry Pi (Fritzing, 2024).

2.2.9 *Confusion Matrix*

Confusion Matrix merupakan sebuah metode yang digunakan untuk menganalisis kinerja sebuah model klasifikasi. Elemen-elemen pada diagonal utama menunjukkan prediksi yang benar, sementara hasil yang salah diklasifikasikan muncul di luar diagonal utama. Oleh karena itu, klasifikasi yang ideal akan menghasilkan *Confusion Matrix* dengan nilai-nilai yang hanya pada diagonal utama, dan semua elemen lainnya bernilai nol. *Confusion Matrix* memberikan informasi tentang nilai aktual dan nilai prediksi setelah proses klasifikasi. Efektivitas dalam sistem ditentukan berdasarkan nilai-nilai yang terkandung dalam matriks. Untuk pengklasifikasi dengan dua kelas, *Confusion Matrix* dapat digambarkan dalam tabel berikut (Arias-Duart dkk., 2023).

Tabel 2.2 *Confusion Matrix*

Matriks		Prediksi Kelas	
		Positive	*Negative*
Kelas Sebenarnya	*Positive*	*True Positive*	*False Negative*
	Negative	*False Positive*	*True Negative*

Confusion Matrix dapat dijelaskan sebagai berikut:

1) *True Positive* (TP): Merupakan jumlah hasil atau prediksi yang benar ketika kelas yang sebenarnya adalah positif.
2) *False Positive* (FP): Merupakan jumlah hasil atau prediksi yang salah ketika kelas yang sebenarnya adalah positif.
3) *True Negative* (TN): Merupakan jumlah hasil atau prediksi yang benar ketika kelas yang sebenarnya adalah negatif.
4) *False Negative* (FN): Merupakan jumlah hasil atau prediksi yang salah ketika kelas yang sebenarnya adalah negatif.

Accuracy dihitung untuk mengevaluasi kinerja sistem dengan mempertimbangkan jumlah total prediksi yang benar yang dibuat oleh pengklasifikasi. Rumus untuk menghitung akurasi bisa dilihat pada persamaan 2.1 adalah sebagai berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Recall dihitung berdasarkan proporsi *input* positif yang teridentifikasi dengan benar. Ini mengukur tingkat TP dan dihitung pada persamaan 2.2 dengan rumus berikut:

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Precision mengukur jumlah kasus positif yang diprediksi dengan benar oleh pengklasifikasi. Rumus untuk menghitung *precision* pada persamaan 2.3 adalah sebagai berikut:

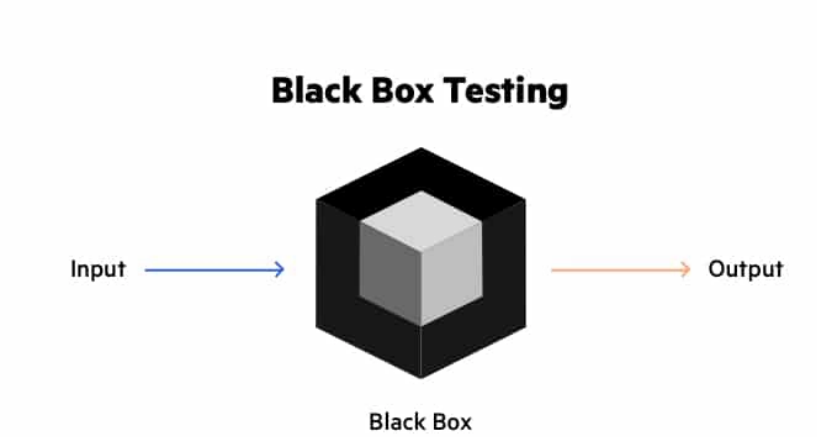
$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

F1-Score merupakan ukuran akurasi yang menggabungkan *precision* dan *recall*. Nilai *F1-Score* berada di antara 0 dan 1, dengan nilai tertinggi di 1 dan terendah di 0. *F1-Score* dihitung pada persamaan 2.4 dengan rumus berikut:

$$F1\ Score = 2\ x \left(\frac{Precision\ x\ Recall}{Precision + Recall}\right) \tag{2.4}$$

2.2.10 *Black Box Testing*

Black Box Testing adalah salah satu metode pengujian perangkat lunak yang sangat umum digunakan, di mana fokus utamanya adalah menguji fungsionalitas sistem tanpa memperhatikan bagaimana struktur internal, arsitektur, atau kode program dari perangkat lunak yang diuji. Pendekatan ini bertujuan untuk menilai apakah sistem berjalan sesuai dengan spesifikasi dan persyaratan yang telah ditentukan tanpa melibatkan pemahaman teknis mendalam tentang bagaimana sistem tersebut dirancang atau diimplementasikan. Dalam *Black Box Testing*, penguji biasanya hanya mengetahui fungsi-fungsi yang ada, dan pengujian dilakukan dengan memberikan *input* tertentu untuk memverifikasi apakah *output* yang dihasilkan sesuai dengan yang diharapkan (Umam dkk., 2023).



Gambar 2.11 *Black Box Testing*
(Sumber: https://bit.ly/4aYpaID)

Pengujian ini juga dikenal sebagai *behavioral testing* yang akan berfokus pada *input* dan *output* untuk memastikan bahwa perangkat lunak dapat berjalan dengan baik. *Behavioral testing* bertujuan untuk memastikan bahwa setiap fungsi perangkat lunak dapat berjalan dengan baik dan mampu memenuhi kebutuhan pengguna. Proses pengujian ini melibatkan berbagai skenario uji, mulai dari pengujian dengan *input* valid untuk memeriksa apakah perangkat lunak memberikan hasil yang benar hingga pengujian dengan *input* tidak valid untuk melihat bagaimana perangkat lunak menangani kesalahan atau anomali.

Keunggulan utama *Black Box Testing* adalah pendekatannya yang sederhana dan tidak memerlukan akses ke kode, sehingga dapat dilakukan oleh penguji *non-programmer* sekalipun. Metode ini juga efektif dalam menemukan kesalahan fungsi yang langsung terlihat dari sisi pengguna. Namun, karena pengujian hanya berfokus pada *input* dan *output*.