BAB 2
LANDASAN TEORI

Terdapat beberapa teori yang digunakan dalam penelitian, yaitu perbedaan daging sapi segar dan beku yang telah dicairkan, *Convolutional Neural Network*, *Transfer Learning*, dan *Confusion Matrix*.

2.1 Perbedaan Daging Sapi Segar dan Beku yang dicairkan

Untuk membedakan daging segar dari daging beku yang telah dicairkan, beberapa hal dapat membantu membedakan keduanya, seperti struktur fisik, warna, dan tekstur. Daging segar biasanya lembut dan elastis, dan daging beku yang telah dicairkan biasanya kasar dan tidak konsisten. Sifat kimia seperti jumlah air, lemak, dan protein yang berbeda antara kedua jenis daging juga dapat berfungsi sebagai pertanda penting untuk membedakan antara keduanya.

2.1.1 Warna Daging

Perbedaan antara daging segar dan daging beku yang dicairkan salah satunya adalah dari warna. Perbedaan warna dari daging sapi segar dan daging sapi beku yang telah dicairkan dapat dilihat pada Gambar 2.1a dan 2.1b.

(a) Daging sapi segar
Sumber: [16]

(b) Daging sapi beku yang telah dicairkan
Sumber: [17]

Gambar 2.1. Perbedaan warna daging sapi segar dan beku yang telah dicairkan

Pada Gambar 2.1a merupakan gambar daging sapi segar. Daging sapi yang baik memiliki warna merah cerah merata [18], mengkilap, lembap, tidak pucat, dan

tidak terdapat noda hitam atau putih pada daging serta warna lemak pada daging berwarna putih ke kuning dan masih terdapat bercak darah.

Warna pada daging beku cenderung berwarna merah gelap keputihan karena tertutup lapisan es, jika daging memiliki banyak lemak terlihat dan warna lemak dagingnya berwarna putih tebal dan kasar.

Sedangkan untuk gambar 2.1b merupakan daging sapi beku yang telah dicairkan. Daging beku yang dicairkan cenderung berwarna merah keabu-abuan, gelap atau ke cokelatan, pucat dan mengerut serta terdapat noda hitam atau putih pada daging. Daging beku yang dicairkan seperti daging yang sedang mengalami pembusukan karena terpengaruh penyerapan bekas air pembekuan dan rusaknya tekstur daging akibat pembekuan [5], [6].

2.1.2 Tekstur Daging

Perbedaan lain antara daging segar dan daging beku yang dicairkan adalah Daging sapi segar cenderung memiliki tekstur yang kenyal dan padat serta tekstur serat daging dapat terlihat dengan jelas dan masih tersusun dengan rapi [6]. Daging sapi beku memiliki tekstur keras dan kasar serta serat daging tidak terlihat dengan jelas karena tertutup lapisan es, sedangkan daging sapi yang sudah dicairkan cenderung memiliki tekstur yang lembut, agak rapuh dan tekstur serat daging tidak begitu terlihat karena mengerut serta tekstur daging mudah hancur karena proses pembekuan dan pencairan [5].

Selain tekstur dan warna, daging dapat dibedakan melalui aroma dan rasa. Aroma dan rasa dari daging segar harum dan alami, Sedangkan pada daging beku yang dicairkan memiliki aroma kurang sedap, cenderung apek, dan rasa yang tawar dikarenakan pada daging beku yang dicairkan banyak menyerap air [6], [10].

2.2 Convolutional Neural Network (CNN)

Artificial Neural Network (ANN) adalah bagian dari *Convolutional Neural Network* (CNN, atau ConvNet), yang biasanya digunakan untuk menganalisis gambar visual [19]. Selain untuk menganalisis gambar CNN juga sering digunakan untuk memproses video [20], klasifikasi gambar [7] , segmentasi gambar [21], dan lainnya. *Convolutional Neural Network* yang pertama kali dikembangkan oleh Fukushima pada tahun 1980 dengan nama *NeoCognitron* adalah algoritma klasifikasi yang berbasis pada pemrosesan gambar dan penglihatan komputer, dan

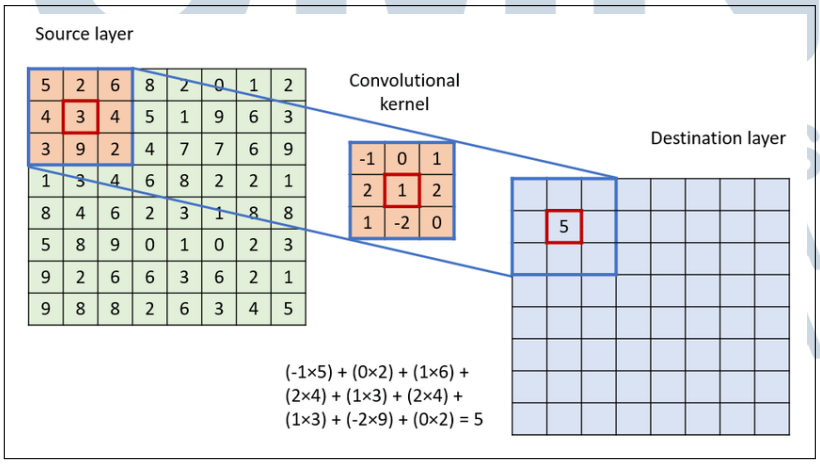
digunakan dalam pembelajaran *deep learning* dengan metode *supervised learning*. CNN terdiri dari berbagai lapisan jaringan dan bekerja dengan cara belajar pola visual yang kuat melalui proses pembelajaran [22]. CNN adalah evolusi dari *Multilayer Perceptron* (MLP) yang memiliki kemampuan untuk mengidentifikasi dan memahami data gambar dua dimensi [23]. Model umum ANN memiliki satu lapisan *input* dan *output* serta beberapa lapisan tersembunyi [24]. Sebuah *neuron* tertentu mengambil vektor *input X* dan menghasilkan *output Y* dengan menjalankan beberapa fungsi F, yang diwakili oleh persamaan umum yang diberikan oleh [25] di bawah.

$$F(X,W) = Y \tag{2.1}$$

Persamaan Rumus 2.1 di atas adalah persamaan pada ANN di mana, W menunjukkan vektor bobot yang merepresentasikan kekuatan interkoneksi antara *neuron* dari dua lapisan yang berdekatan. Vektor bobot yang diperoleh sekarang dapat digunakan untuk melakukan klasifikasi gambar. CNN adalah model yang mendapatkan perhatian karena kemampuan klasifikasinya berdasarkan informasi kontekstual. Model umum CNN terdiri dari empat komponen yaitu *convolution layer*, *pooling layer*, fungsi *activation*, dan *fully connected layer*.

2.2.1 Convolutional Layer

Bagian penting dari proses CNN, dengan operasi konvolusi yang mengarahkan setiap filter ke bagian lain dari *pixel* berdasarkan ukuran *kernel* yang digunakan.

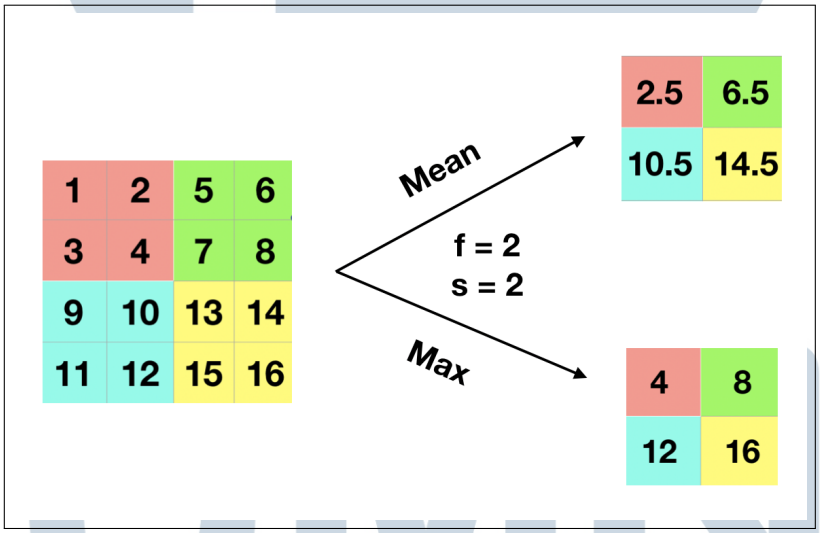


Gambar 2.2. *Convolutional layer*

Seperti yang ditunjukkan pada Gambar 2.2 , setiap filter melakukan perhitungan antara nilai *input* (*pixel* gambar) dan nilai pada filter setelah itu filter mengalami pergeseran untuk melakukan perhitungan pada *pixel* di filter selanjutnya sampai merata. *Pixel* yang telah dikalikan akan menghasilkan nilai *output*. Proses mirip seperti perkalian matriks.

2.2.2 Pooling Layer

Pooling layer merupakan lapisan berisi nilai *output* dari tahapan konvolusi yang digunakan. Ukuran *stride* digunakan pada *layer* untuk menjaga filter bergerak di seluruh luas *pixel feature map*. *Pooling* bertujuan untuk mengurangi kompleksitas komputasi di dalam jaringan dengan meningkatkan skala spasial deskripsi gambar secara perlahan.



Gambar 2.3. *Pooling layer*

Seperti Pada Gambar 2.3, F merupakan singkatan dari Filter dan S merupakan singkatan dari *Stride*. Pada proses *max pooling* pada gambar awalnya memiliki nilai *input* berukuran 4 x 4 yang setiap empat angka pada setiap kotak dengan warna berbeda yang diambil angka maksimalnya lalu dijadikan *output* dengan ukuran Filter 2x2 dengan perpindahan sebanyak dua kolom atau dua *stride*, Begitu juga dengan proses *pooling* lainnya.

Terdapat beberapa jenis *pooling* yang dapat digunakan, termasuk *maximum pooling*, *minimum pooling* ,*average pooling* ,dan *adaptive pooling*. *Max-pooling* adalah jenis yang paling umum digunakan dan membantu menjaga fitur yang

dominan dengan mempertahankan rotasi dan translasi yang sama dalam pelatihan model yang efisien. Sementara itu, *average pooling* dapat memberikan hasil yang baik dalam mengurangi *noise* di gambar. Keduanya dapat membantu dalam pengurangan dimensi gambar dan mengoptimalkan kinerja CNN [26]

2.2.3 Flatten Layer

Setelah serangkaian lapisan konvolusi dan *pooling*, hasilnya harus diubah menjadi vektor satu dimensi sebelum dimasukkan ke dalam *Fully Connected Layer*. Lapisan *Flatten* digunakan untuk melakukan dengan mengubah setiap peta fitur menjadi vektor satu dimensi.

2.2.4 Dropout Layer

Untuk mencegah *overfitting*, *dropout layer* diterapkan setelah lapisan-lapisan konvolusi dan *Fully Connected Layer*. *Dropout* secara acak mengabaikan sebagian unit selama pelatihan, sehingga memaksa jaringan untuk belajar fitur dan tidak tergantung pada fitur spesifik dari sampel pelatihan.

2.2.5 Fully Connected Layer

Dengan memproses hasil akhir dari operasi *convolution* dan *pooling*, *Fully-Connected Neural Network* membuat prediksi, menemukan label yang paling sesuai untuk mewakili gambar. Jaringan menggunakan bobot yang dihitung selama pelatihan untuk menghubungkan vektor fitur gambar dan kelas gambar. *Output* dari *convolution* dan *pooling* dikalikan dengan bobot yang sesuai, dan kemudian diproses melalui fungsi *activation* [27]. *Neuron* pada lapisan berhubungan dengan *neuron* pada lapisan sebelumnya dan sesudahnya [28].

2.2.6 Fungsi Aktivasi

Aktivasi dapat digunakan melalui lapisan, atau melalui argumen yang didukung oleh semua lapisan lanjutan Aktivasi. Banyak penelitian menunjukkan bahwa fungsi *activation softmax* digunakan dalam algoritma pembelajaran mesin konvensional. Berikut beberapa penjelasan mengenai *activation* yang digunakan.

- Rectified Linear Unit (ReLU)
  Penggunaan *Rectified Linear Unit* (ReLU) telah terbukti efektif

memperkenalkan non-linearitas [29]. perhitungan turunan parsial ReLU mudah dilakukan. Berikut persamaan *activation Rectified Linear Unit* (ReLU).

$$Fx = Max(0, x) \tag{2.2}$$

Pada persamaan Rumus 2.2 Jika (x) positif atau nol, (f(x)) mengembalikan nilai (x). Namun jika (x) negatif, (f(x)) mengembalikan nilai nol. Dengan ReLU dapat menghilangkan bias pada gambar dan memperjelas fitur garis putih pada gambar agar model dapat menangkap fitur pada gambar dengan baik. ReLU memungkinkan jaringan saraf memodelkan hubungan non-linear antara fitur untuk membantu mempelajari representasi yang kompleks dan gradien tidak dapat dihilangkan oleh ReLU.

- Softmax

  Fungsi *activation softmax* dikembangkan untuk tugas klasifikasi multi kelas di mana *input* harus diberi label ke salah satu dari beberapa kelas. Fungsi mengubah skor mentah, yang biasanya disebut sebagai *logit*, menjadi distribusi probabilitas atas beberapa kelas, yang berarti fungsi memberikan probabilitas untuk setiap kelas, menunjukkan seberapa mungkin *input* termasuk dalam kelas tersebut. Berikut rumus matematika untuk fungsi *activation softmax*.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.3}$$

Pada persamaan Rumus 2.3, $\sigma(z_i)$ adalah probabilitas untuk kelas ke i. z_i adalah nilai *logit* dari elemen ke i dari *input vector* **z**. $\sum_j e^{z_j}$ adalah jumlah dari semua eksponensial dari elemen-elemen *vector input* **z**. Pada *output* dari persamaan *softmax* akan menampilkan *output* berupa probabilitas dari setiap kelas atau elemen. Probabilitas dapat diinterpretasikan sebagai kemungkinan dari setiap kelas. Probabilitas tertinggi menunjukkan prediksi kelas model. Fungsi melakukan konversi *logits* atau nilai keluaran dari *layer* sebelumnya menjadi probabilitas dari setiap kelas yang jika dijumlahkan hasilnya adalah satu.

2.3 Transfer Learning

Transfer learning adalah metode yang ampuh yang memanfaatkan informasi yang diperoleh dari satu domain untuk meningkatkan kinerja pada tugas atau domain yang terpisah tetapi terkait [30]. situasi sangat berguna ketika mendapatkan data yang berlabel sulit atau mahal. *Transfer Learning* memfasilitasi pembuatan model yang sangat akurat dan efektif dengan waktu pelatihan yang singkat dengan mentransfer representasi dan informasi yang diperoleh dari *pre trained model* [31].

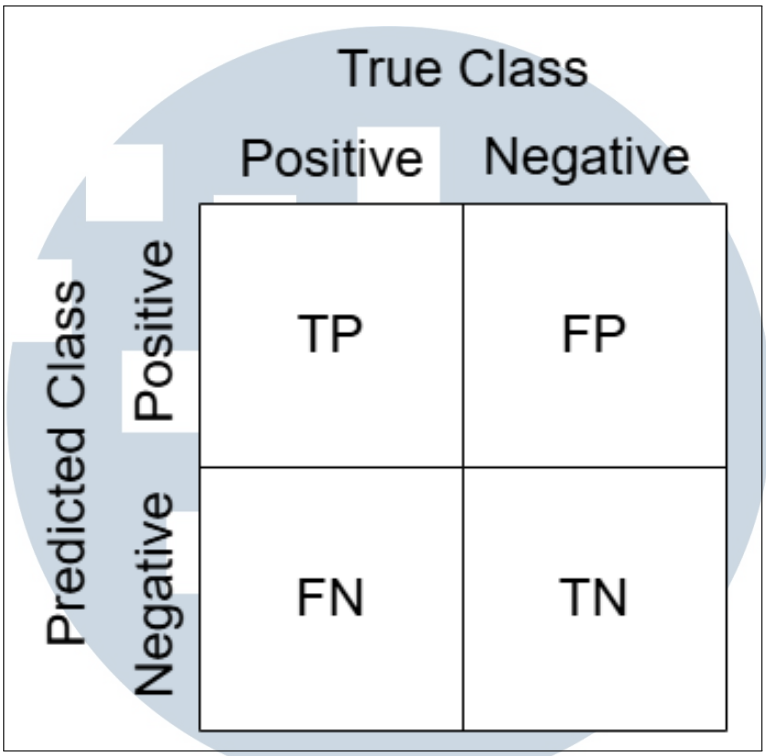
Transfer Learning telah menarik banyak minat dan telah menunjukkan cukup sukses di bidang pencitraan gambar. Namun *pre trained model* atau *Transfer Learning* model jarang digunakan pada bidang pencitraan daging. Model CNN yang juga dikenal sebagai ConvNet yang telah dilatih pada *dataset* skala besar, model seperti VGG, ResNet, AlexNet, DenseNet [32] dan lainnya. *Transfer Learning* digunakan untuk menangkap fitur visual kaya yang relevan di berbagai tugas pengenalan gambar [33]. Model *pre-trained* telah menguasai kemampuan untuk mengenali dan mengekstrak karakteristik tingkat tinggi seperti pola dan struktur objek serta elemen tingkat rendah seperti tepi, tekstur, warna dan bentuk [34].

Transfer learning menunjukkan potensi yang cukup besar untuk deteksi dan prediksi [30]. Karena *dataset* daging yang dibutuhkan sulit untuk dicari dan temukan sehingga menggunakan *dataset* yang tersedia ada sekarang tidak bisa melatih seluruh jaringan *convolution* dari awal serta keterbatasan waktu dalam melakukan penelitian. Pilihan lain adalah menggunakan ConvNet yang telah dilatih pada *dataset* besar seperti *ImageNet* [35], sebagai inisialisasi atau ekstraksi fitur tetap untuk tugas yang ada. menggunakan Jaringan *convolutional pretrained* yang telah dilatih pada *dataset* yang cukup besar seperti *ImageNet* [35] dari pada membuat jaringan *convolutional* dari awal.

2.4 Confusion Matrix

Confusion matrix merupakan alat untuk analisis hasil prediksi di *machine learning* [36]. *Confusion matrix* berupa sebuah tabel dari beberapa jumlah prediksi yang benar dan salah yang dihasilkan oleh klasifikasi atau model klasifikasi untuk tugas klasifikasi biner [36]. *Confusion marix* merupakan matriks N x N yang dipakai untuk mengevaluasi kinerja model klasifikasi, di mana N adalah jumlah target kelas. Dengan visualisasi matriks, dapat dengan mudah menentukan akurasi

model dengan mengamati nilai diagonal pengukuran jumlah klasifikasi yang akurat.



Gambar 2.4. *Confusion matrix*

Pada Gambar 2.4, *True Positive* (TP) adalah jumlah *instance* yang benar positif dan diprediksi positif oleh model. *True Negative* (TN) adalah jumlah *instance* yang benar negatif dan diprediksi negatif oleh model. *False Positive* (FP) adalah jumlah *instance* yang benar negatif tetapi diprediksi positif oleh model (*type I error*). *False Negative* (FN) adalah jumlah *instance* yang benar positif tetapi diprediksi negatif oleh model (*type II error*). Selanjutnya dari *Confusion Matrix* berguna untuk mengukur *classification report* seperti *accuracy, precision, recall* dan F1 *score*. Berikut persamaannya.

- Accuracy
  Akurasi mengukur proporsi prediksi benar yang dihasilkan oleh model berdasarkan *dataset* uji [36]. Akurasi merupakan metrik dasar yang baik untuk menilai performa model. Namun, pada *dataset* yang tidak seimbang, akurasi dapat menjadi metrik yang kurang representatif dan kurang informatif mengenai performa model. Berikut persamaan untuk *accuracy*.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.4}$$

Pada persamaan Rumus 2.4, seluruh prediksi benar dari *True Potisitive* (TP) dan *True Negative* (TN) dibagi dengan keseluruhan prediksi yaitu *True Positive* (TP) + *True Negative* (TN) + *False Positive* (FP) + *False Negative* (FN).

- Precision

  Presisi memberitahu seberapa banyak kasus yang diprediksi dengan benar sebenarnya ternyata positif. Dapat menentukan apakah model yang dibuat dapat diandalkan atau tidak. Presisi secara singkat di definisikan sebagai perbandingan antara *True Positive* (TP) dengan banyaknya data yang diprediksi positif. Berikut persamaan matematisnya.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.5}$$

  Pada persamaan Rumus 2.5, proporsi prediksi positif yang benar positif (*True positive*) dibagi dengan total jumlah proporsi prediksi positif yang benar positif (*True Positive*) dan proporsi prediksi positif yang salah negatif (*False Negative*).

- *Recall* (Sensivity)

  Recall memberitahu seberapa banyak hasil *Positive* sebenarnya yang dapat diprediksi oleh model [36]. Secara definisi *recall* adalah perbandingan antara *True Positive* (TP) dengan banyaknya data yang sebenarnya positif. Berikut persamaan matematisnya.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.6}$$

  Pada persamaan Rumus 2.6, proporsi *instance* positif yang diprediksi positif (*True Positive*) dibagi dengan total jumlah proporsi *instance* positif yang diprediksi positif (*True Positive*) dan proporsi *instance* negatif yang diprediksi positif (*False Negative*).

- F1 Score

  Ketika upaya dilakukan untuk meningkatkan presisi model, *recall* cenderung menurun, dan sebaliknya [36]. Meskipun terdapat situasi ideal di mana data dapat dipisahkan secara sempurna, mencapai skor 1.0 untuk keduanya sangat jarang, bahkan hampir tidak pernah terjadi dalam kehidupan nyata. Oleh

karena itu, saat memilih model, sebaiknya tidak mengandalkan skor tunggal di antara keduanya. Skor yang lebih representatif adalah f1 *score*. F1 *score* adalah rata-rata dari *accuracy*, *precison* dan *recall*, memberikan gambaran kombinasi dari kedua metrik tersebut. Nilainya mencapai maksimum ketika presisi sama dengan *recall*. Berikut persamaan matematisnya.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}$$

Pada persamaan Rumus 2.7, matematis f1 *score* adalah dua dikalikan proporsi *Precision* dikali dengan proporsi *Recall* dibagi dengan total jumlah proporsi *Precision* dengan proporsi *Recall*.

2.5 Augmentasi Data

Data augmentation atau augmentasi data merupakan teknik yang digunakan dalam pembelajaran mesin dan pengolahan citra untuk meningkatkan jumlah dan keragaman data pelatihan tanpa harus mengumpulkan data baru [37]. Teknik ini mencakup berbagai metode seperti transformasi geometris (rotasi, translasi, skala), perubahan warna, *flipping* (membalik gambar) dan *cropping* (memotong gambar). Salah satu augmentasi gambar dapat dilakukan dengan menggunakan *ImageDataGenerator* API dari *TensorFlow Keras preprocessing image* [38], [39].

ImageDataGenerator dapat digunakan secara langsung untuk menghasilkan kumpulan gambar yang berbeda di setiap *batch* saat melatih model [40]. Tujuan *ImageDataGenerator* agar model dapat melihat gambar yang sedikit berbeda di setiap *batch* dan dapat meningkatkan efisiensi model [40].

ImageDataGenerator menyediakan banyak parameter untuk menambah variasi data yang diinginkan. Seperti pada dokumentasi tutorial *website* Keras mengenai membangun model klasifikasi gambar yang kuat dengan menggunakan data yang sangat sedikit menggunakan *ImageDataGenerator* [41].

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
```

```
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

Kode 2.1: Kode konfigurasi ImageDataGenerator pada dokumentasi tutorial Keras

Pada Kode 2.1 dokumentasi tutorial Keras tersebut *ImageDataGenerator* menggunakan konfigurasi rotasi sebesar 40 derajat, *widith shift range* dan *height shift range* sebesar 20%, *shear range* sebesar 20%, pembesaran gambar sebesar 20%, dengan penambahan dibalik horizontal, untuk bagian yang tidak terkena gambar ditambahkan *fill mode* untuk mengisi gambar yang kosong dengan warna terdekat dengan gambar aslinya menggunakan *nearest*. Konfigurasi ini dapat menghasilkan *accuracy* 80% sampai 94% [41].