PENERAPAN ALGORITMA *CANNY* DAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION* DALAM MENGIDENTIFIKASI IKAN KOI

Robby Zain Zakaria, Haris Yuana, Udkhiati Mawaddah
Teknik Informatika, Universitas Islam Balitar
Jalan Majapahit No 4 Kec. Sanan Wetan Blitar, Indonesia
robbyzain13@gmail.com

ABSTRAK

Penelitian ini bertujuan menerapkan Algoritma *Canny* dan Jaringan Syaraf Tiruan *Backpropagation* dalam proses identifikasi Ikan Koi. Algoritma *Canny* digunakan untuk pengolahan citra, mengubah citra Ikan Koi menjadi citra biner guna mendeteksi tepi dan fitur utama. Sementara itu, Algoritma *Backpropagation*, yang merupakan algoritma pembelajaran terawasi dengan struktur *multilayer*, digunakan untuk melatih dan menguji model pengenalan Ikan Koi. Hasil pengujian menunjukkan tingkat keberhasilan yang memuaskan dalam mengenali jenis-jenis Ikan Koi yang berbeda. Dengan menguji model pada citra latih sebanyak 45, tingkat keberhasilan dikenali mencapai 63,33%, dengan 60 citra latih mencapai 66,67%, dan dengan 90 citra latih mencapai 70%. Hasil ini memberikan wawasan penting mengenai potensi teknologi dalam identifikasi Ikan Koi serta potensi peningkatan akurasi melalui pengembangan lebih lanjut.

Kata Kunci: *Algoritma Canny, Akurasi Pengenalan, Ikan Koi, Jaringan Syaraf Tiruan Backpropagation, Pengolahan Citra.*

1. PENDAHULUAN

Perlombaan Ikan Koi berkualitas sangat banyak diminati oleh berbagai kalangan masyarakat maupun pembudidaya Ikan Koi, dikarenakan bentuk tubuh dan pola dari Ikan Koi tersebut sangat indah. Ikan Koi atau *Nishikigoi* berasal dari *Niigata*, salah satu prefektur di Jepang. Istilah *Nishikigoi* sendiri sudah ada sejak 200 tahun yang lalu. Ikan Koi diproduksi oleh petani yang membudidayakan ikan mas hitam sebagai sumber makanan untuk bertahan hidup dalam kondisi cuaca musim dingin yang parah. Hasil yang lahir dari ikan mas tersebut adalah ikan mas berwarna cerah dengan sosok mengagumkan yang menonjol dari ikan-ikan yang lainnya. Seiring waktu, banyak yang mulai mengapresiasi Ikan Koi seperti sebuah karya seni yang indah.

Ikan Koi menjadi objek yang menarik untuk diteliti karena memiliki pola dan warna yang kompleks. Keunikan pola dan warna Ikan Koi membuatnya sulit untuk diidentifikasi dan dibedakan antara satu jenis dengan yang lain. Penulis akan mengklasifikasikan kategori Ikan Koi dengan Algoritma *Canny* dan Jaringan Syaraf Tiruan *Backpropagation*. Algoritma *Canny* digunakan untuk mendeteksi tepi pada gambar Ikan Koi. Tepi ini merupakan fitur penting yang membantu memisahkan objek (Ikan Koi) dari latar belakang dalam gambar. Algoritma *Canny* menggunakan pendekatan multi-tahap yang mencakup proses *smoothing*, perhitungan gradien, *non-maximum suppression*, dan *hysteresis thresholding* untuk menghasilkan tepi yang akurat dan tajam. Sedangkan, Jaringan Syaraf Tiruan *Backpropagation* digunakan untuk melatih dan menguji model jaringan syaraf tiruan menggunakan data training yang telah diproses. Dalam konteks ini, jaringan syaraf tiruan akan belajar mempelajari pola dan fitur dari gambar Ikan Koi yang telah diekstraksi sebelumnya.

2. TINJAUAN PUSTAKA

2.1. Pengolahan Citra

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan. Suatu citra dapat didefinisikan sebagai fungsi f (x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan *amplitude* f di titik koordinat x,y dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut.

Citra analog adalah citra yang bersifat kontinu, seperti gambar pada monitor televisi, foto sinar X, foto yang tercetak di kertas foto, lukisan, pemandangan alam, hasil *CT scan*, gambar-gambar yang terekam pada pita kaset, dan lain sebagainya. Citra analog tidak dapat dipresentasikan dalam komputer sehingga tidak bisa diproses secara langsung. Oleh sebab itu, agar citra ini dapat diproses di komputer, proses konversi analog ke digital harus dilakukan terlebih dahulu.

Secara umum, pengolahan citra digital menunjukan pada pemrosesan gambar dua dimensi menggunakan komputer. Dalam konteks yang lebih luas pengolahan citra digital mengacu pada pemrosesan setiap data dua dimensi. Citra digital merupakan sebuah larih (*array*) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu [1].

2.2. Citra *Grayscale*

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya, dengan kata lain nilai bagian *RED = GREEN = BLUE*. Nilai tersebut digunakan untuk menulnjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekatl putih. Citra *grayscale* berikut memiliki kedalaman warna 8 bit (255 kombinasi warna keabuan). [2]

Sedangkan, *Grayscaling* adalah suatu proses mengubah citra digital menjadi citra hitam pultilh (*grayscale*) dengan menghilangkan informasi warna dari setiap piksel. Citra hasilnya terdiri dari berbagai gradasi warna abu-abu (*grayscale*) mulai dari hitam pekat hingga putih bersih, yang merepresentasikan intensitas cahaya pada setiap piksel citra asli. Proses *Grayscaling* dapat dilakukan dengan mengambil rata-rata nilai dari setiap kanal warna pada setiap piksel citra asli dan menggantinya dengan nilai keabuan yang sesuai. Konversi gambar normal menjadi gambar *grayscale* dapat dilakukan menggunakan rumus:

$$Gray = \frac{(R+G+B)}{3} \qquad (1)$$

Keterangan:
R = *Red*
G = *Green*
B = *Blue*

2.3. Algoritma *Canny*

Salah satu algoritma deteksi tepi modern adalah deteksi tepi dengan menggunakan metode *Canny*. Deteksi tepi *Canny* ditemukan oleh Marr dan Hildreth yang meneliti pemodelan persepsi visual manusia. Ada beberapa kriteria pendeteksi tepian paling optimum yang dapat dipenuhi oleh algoritma *Canny*:

a. Mendeteksi dengan baik (kriteria deteksi)
   Kemampuan untuk meletakkan dan menandai semua tepi yang ada sesuai dengan pemilihan parameter-parameter konvolusi yang dilakukan. Sekaligus juga memberikan fleksibilitas yang sangat tinggi dalam hal menentukan tingkat deteksi ketebalan tepi sesua yang diinginkan.
b. Melokalisasi dengan baik (kriteria lokalisasi)
   Dengan *Canny* dimungkinkan dihasilkan jarak yang minimum antara tepi yang dideteksi dengan tepi yang asli.
c. Respon yang jelas (kriteria respon)
   Hanya ada satu respon untuk tiap tepi. Sehingga mudah dideteksi dan tidak menimbulkan kerancuan pada pengolahan citra selanjutnya.

Pemilihan parameter deteksi tepi *Canny* sangat mempengaruhi hasil dari tepian yang dihasilkan. Beberapa parameter tersebut antara lain:

1. Nilai Standart Deviasi *Gaussian*
2. Nilai Ambang

Pendekatan algoritma *canny* dilakukan dengan konvolusi fungsi gambar dengan operator *gaussian* dan turunan-turunannya. Turunan pertama dari fungsi citra yang dikonvolusikan dengan fungsi *gaussian*,

$$g(x,y) = D[gauss(x,y) * f(x,y)] \quad (2)$$

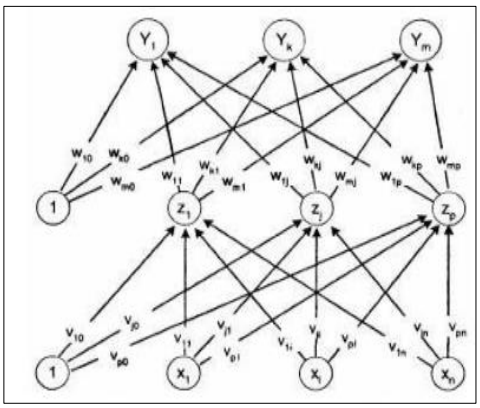
Ekuivalen dengan fungsi citra yang dikonvolusikan dengan turunan pertama dari fungsi *gaussian*,

$$g(x,y) = D[gauss(x,y)] * f(x,y) \quad (3)$$

Oleh karena itu, memungkinkan untuk mengkombinasikan tingkat kehalusan dan pendektesian tepi ke dalam suatu konvolusi dalam satu dimensi dengan dua arah yang berbeda (vertikal dan horizontal). [3]

2.4. Jaringan Syaraf Tiruan *Backpropagation*

Jaringan Syaraf Tiruan *Backpropagation* memiliki beberapa unit yang ada di dalam satu atau lebih *layer* tersembunyi. Gambar berikut adalah arsitektur *Backpropagation* dengan n buah masukan (ditambah sebuah bias), sebuah *layer* tersembunyi yang terdiri dari p unit (ditambah sebuah bias), dan m unit keluaran.



Gambar 1. Arsitektur jst *backpropagation*

Gambar 1 menunjukkan arsitektur jaringan syaraf tiruan *backpropagation*. Pada gambar tersebut, Vji merujuk pada bobot yang menghubungkan unit masukan Xi ke unit di lapisan tersembunyi Vj, di mana Vj0 adalah bobot yang menghubungkan bias ke unit masukan di lapisan tersembunyi Zj. Wkj adalah bobot yang menghubungkan unit di lapisan tersembunyi Zj ke unit keluaran Yk, dan Wko adalah bobot yang menghubungkan bias ke unit keluaran Zk. Struktur ini adalah bagian penting dari algoritma Backpropagation dalam JST dan memainkan peran kunci dalam pembelajaran dan pengenalan pola.. [4]

3. METODE PENELITIAN

Penelitian ini bertujuan untuk menggambarkan dan menganalisis citra Ikan Koi pada *dataset* menggunakan algoritma *canny* dan jaringan syaraf tiruan *backpropagation*. Metode penelitian yang

digunakan adalah deskriptif kuantitatif, yang bertujuan untuk mengumpulkan dan menganalisis data numerik yang dapat diukur secara objektif. Data diperoleh melalui wawancara dengan seorang pembudidaya Ikan Koi yang juga anggota APKI (Asosiasi Pecinta Koi Indonesia) di Desa Butun, Gandusari, Kab. Blitar, serta pengumpulan *dataset* dari internet. Proses pra-pemrosesan data melibatkan perubahan ukuran citra menjadi 800 x 1296 piksel dan konversi citra dari RGB ke *grayscale*. Selain itu, data Ikan Koi dibagi menjadi data latih dan data uji.

Gambar 2. Alur penelitian

Pada Gambar 2 merupakan alur dari penelitian penerapan algoritma *canny* dan jaringan syaraf tiruan *backpropagation* dalam mengidentifikasi Ikan Koi. Berikut adalah penjelasannya:

3.1. Menentukan Topik

Pemilihan topik penelitian terkait algoritma *canny* dan jaringan syaraf tiruan *backpropagation* dalam identifikasi Ikan Koi.

3.2. Identifikasi dan Perumusan Masalah

Mengidentifikasi kesulitan dalam mengidentifikasi pola Ikan Koi secara akurat dan efisien sebagai masalah penelitian.

3.3. Menentukan Tujuan dan Batasan Masalah

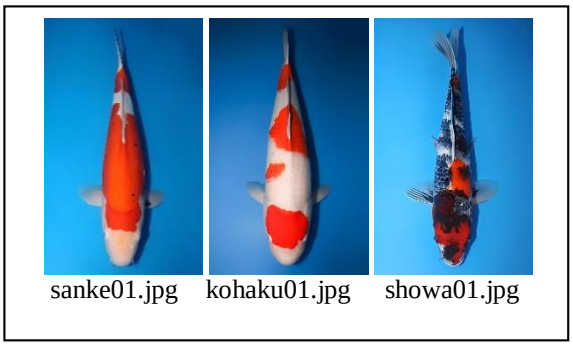
Menetapkan tujuan penelitian untuk meningkatkan efisiensi dan akurasi identifikasi Ikan Koi dengan menggabungkan algoritma *canny* dan jaringan syaraf tiruan *backpropagation*. Menetapkan batasan penelitian, termasuk *dataset*, format gambar, dan perangkat yang digunakan.

3.4. Studi Pustaka

Melakukan penelitian terkait penggunaan algoritma *canny* dan jaringan syaraf tiruan *backpropagation* dalam pengenalan pola dengan mengkaji literatur dan sumber terkait.

3.5. Pengumpulan Data

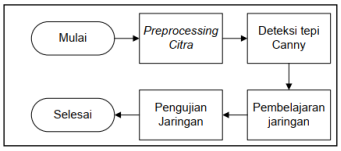
Mengumpulkan *dataset* gambar Ikan Koi yang mencakup jenis dan pola. Menganotasi *dataset* dengan label yang sesuai untuk identifikasi pola Ikan Koi, dengan total 120 gambar Ikan Koi, terdiri dari 90 data latih dan 30 data uji. Data Latih digunakan untuk melatih model dan inisialisasi parameter. Data Uji digunakan untuk menguji performa model dan penyesuaian parameter jika diperlukan. Jenis ikan yang digunakan pada penelitian adalah *Sanke*, *Kohaku* dan *Showa*. Ikan koi *Sanke*, *Kohaku*, dan *Showa* adalah varietas koi yang berbeda dalam hal pola warna dan penampilan. *Sanke* memiliki dasar putih dengan bintik merah dan hitam. *Kohaku* adalah varietas dengan pola putih dengan bintik merah yang lebih sederhana. *Showa*, di sisi lain, memiliki warna dasar hitam dengan bintik merah dan putih. Terlihat pada Gambar 3 *Kohaku* memiliki pola warna paling sederhana, sedangkan *Showa* memiliki warna dasar hitam yang kontras dan *Sanke* memiliki dasar putih dengan tambahan warna merah dan hitam yang lebih mencolok.



Gambar 3. Jenis Ikan Koi yang digunakan pada penelitian

3.6. Perancangan Sistem

Merancang sistem yang menggunakan algoritma *canny* dan jaringan syaraf tiruan *backpropagation* untuk mengidentifikasi pola Ikan Koi. Berikut adalah alur perancangan sistem:



Gambar 4. Alur perancangan sistem

Keterangan:

1. Mulai: Pada tahap ini, sistem dimulai dengan mempersiapkan semua elemen yang diperlukan, seperti perangkat keras dan perangkat lunak yang dibutuhkan.
2. *Preprocessing* Citra: Tahap ini melibatkan pengolahan awal citra yang akan digunakan dalam sistem. *Preprocessing* mencakup langkah-langkah seperti pengurangan *noise*, *smoothing*, dan penyesuaian ukuran gambar.

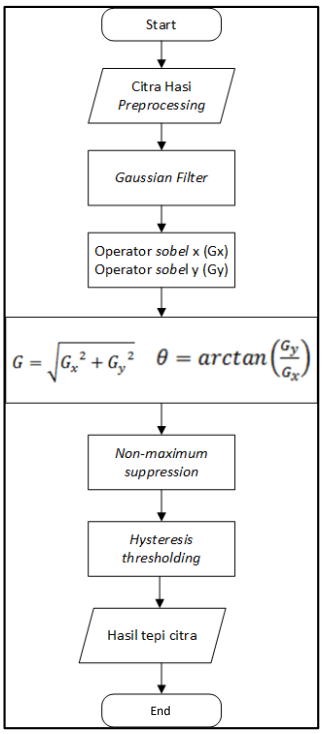
3. Deteksi Tepi *Canny*: Deteksi tepi *canny* adalah teknik yang umum digunakan untuk mengidentifikasi tepi pada citra. Dalam tahap ini, metode deteksi tepi *canny* diterapkan pada citra yang telah melalui tahap *preprocessing* sebelumnya. Hal ini bertujuan untuk menemukan tepi yang signifikan dalam citra.
4. Pembelajaran Jaringan: Pada tahap ini, jaringan syaraf tiruan *backpropagation* dilatih menggunakan *dataset* yang telah dianotasi. Proses pembelajaran ini bertujuan untuk mengajarkan jaringan untuk mengenali pola-pola dan fitur-fitur yang relevan dalam citra.
5. Pengujian Jaringan: Setelah jaringan telah dilatih, tahap pengujian dilakukan untuk mengevaluasi performa jaringan. Pada tahap ini, citra-citra uji yang tidak digunakan dalam proses pembelajaran diberikan kepada jaringan untuk diuji kinerjanya. Hal ini memberikan pemahaman tentang sejauh mana jaringan mampu mengenali objek atau fitur yang diinginkan.
6. Selesai: Setelah melalui tahap *preprocessing*, deteksi tepi, pembelajaran jaringan, dan pengujian, alur sistem dianggap selesai. Pada titik ini, hasil-hasil yang diperoleh dapat dievaluasi, dianalisis, dan digunakan untuk tujuan tertentu sesuai dengan kebutuhan aplikasi yang digunakan.

Setelah dilakukan perancangan alur, kemudian dilakukan implementasi sistem. Berikut adalah perangkat yang digunakan pada penelitian:

1. *Hardware*
   a. *Processor: 11th Gen Intel(R) Core(TM) i5-11400H*
   b. *VGA: NVIDIA GeForce RTX 3050*
   c. *Memory: 24 GB (8 GB onboard and 16 GB add-on)*
2. *Software*
   a. *Operating System: Windows 11 Home Single Language (64-bit, x64-based processor)*
   b. *Software Tools: MATLAB R2021a, Adobe Photoshop 2020*
3. *Dataset*
   a. Jenis Ikan Koi yang digunakan untuk penelitian: *Sanke*, *Kohaku*, dan *Showa*
   d. Dimensi gambar: 800 x 1296 piksel (melalui tahap *preprocessing*)
   e. Sumber gambar: Pengambilan gambar menggunakan kamera *smartphone* (25 MP)

3.7. Deteksi Tepi Objek dengan Algoritma *Canny*

Menerapkan algoritma *canny* pada *dataset* gambar Ikan Koi untuk mendeteksi tepi objek dan memperbaiki detail citra yang kabur karena *error* atau adanya efek akuisisi.



Gambar 5. *Flowchart* deteksi tepi objek menggunakan Algoritma *Canny*

Berikut adalah penjelasan dari Gambar 5, yaitu proses deteksi tepi objek menggunakan Algoritma *Canny*:

1. Start: Proses deteksi Canny dimulai dengan citra hasil *preprocessing* yang telah dikonversi menjadi citra *grayscale*. Input yang digunakan adalah citra Ikan Koi dalam format RGB. Dalam tahap *preprocessing*, citra RGB tersebut diubah menjadi citra keabuan menggunakan proses *grayscaling* sebagai masukan untuk proses *canny*. Proses konversi *grayscale* dilakukan dengan menggunakan persamaan. Setiap piksel gambar diekstraksi untuk mendapatkan nilai R, G, dan B. Setelah itu, nilai-nilai tersebut dijumlahkan dan dibagi 3. Sebagai contoh, misalkan nilai R (1,1) = 93, nilai G (1,1) = 83, nilai B (1,1) = 73.

$$Gray = \frac{(93 + 83 + 73)}{3}$$

2. *Gaussian Filter*: Langkah selanjutnya adalah menerapkan filter *Gaussian* pada citra *grayscale*. Filter ini berfungsi untuk mengurangi *noise* dan memperhalus citra. Dalam proses penghalusan terlebih dahulu kernel dirancang berdasarkan pada ordo matriks m x n dan nilai standar deviasi menggunakan persamaan kemudian dikonvolusi dengan matriks citra. Pada penelitian ini menggunakan matriks dengan ordo 7x7, dikarenakan mencakup piksel-piksel sekitar titik tengah dengan cukup lebar dan menggunakan sigma 2 dalam persamaan *Gaussian* yang berfungsi untuk mengendalikan tingkat

penghalusan serta kepekaan terhadap detail dalam citra. Nilai sigma yang lebih besar akan menghasilkan penghalusan yang lebih signifikan.

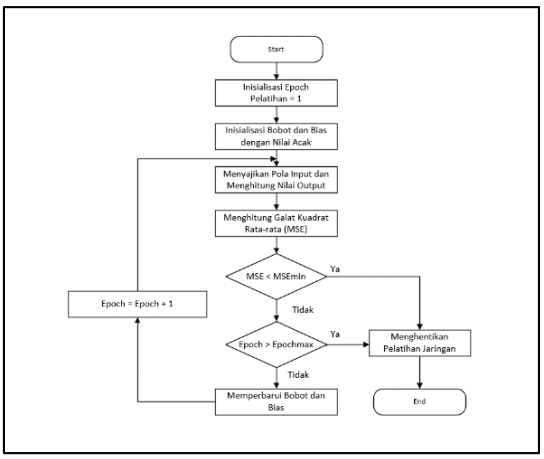
3. Deteksi menggunakan operator *Sobel*: Setelah diterapkan filter *Gaussian*, operator *Sobel* digunakan untuk mendeteksi gradien citra. Operator *Sobel* terdiri dari dua kernel, yaitu kernel *Sobel* horizontal (Gx) dan kernel *Sobel* vertikal (Gy). Kedua kernel ini digunakan untuk menghitung gradien horizontal dan gradien vertikal di setiap piksel citra.
4. Deteksi gradien dengan rumus *arctan*: Gradien horizontal dan vertikal yang telah dihitung dengan operator *Sobel* digunakan untuk mencari gradien magnitude dan arahnya. Rumus *arctan* digunakan untuk menghitung arah gradien pada setiap piksel citra.

$$\theta = arctan\frac{\mathrm{Gy}}{\mathrm{Gx}} \qquad (4)$$

5. *Non-maximum suppression*: Langkah selanjutnya adalah menerapkan *non-maximum suppression* untuk menghilangkan piksel yang bukan merupakan tepi tajam. Proses ini melibatkan perbandingan piksel gradien dengan piksel sekitarnya dalam arah gradien. Jika piksel gradien merupakan maksimum lokal dalam arah gradien, maka piksel tersebut tetap dipertahankan sebagai tepi, sedangkan piksel yang tidak merupakan maksimum lokal akan dihapus.
6. *Hysteresis thresholding*: Tahap ini melibatkan penerapan *thresholding* ganda (*hysteresis*) untuk mengkategorikan tepi menjadi tepi kuat, tepi lemah, atau bukan tepi. Tepi kuat adalah piksel yang memiliki gradien yang cukup besar dan jelas merupakan tepi, sedangkan tepi lemah adalah piksel yang memiliki gradien lebih rendah dan mungkin merupakan tepi atau noise. Piksel bukan tepi adalah piksel yang memiliki gradien yang rendah dan dianggap bukan tepi. *Thresholding* ganda menggunakan batas atas (*upper threshold*) dan batas bawah (*lower threshold*) untuk mengkategorikan piksel menjadi tepi kuat (*strong edges*), tepi lemah (*weak edges*), atau bukan tepi. Dalam penelitian ini menggunakan nilai batas atas 45 dan nilai batas bawah 5, tepi dengan *magnitude* gradien di bawah 5 akan diabaikan, sedangkan tepi dengan *magnitude* gradien di atas 45 akan dianggap sebagai tepi yang kuat. Tepi dengan *magnitude* gradien antara 5 dan 45 akan dianggap sebagai tepi yang lemah, tetapi tetap dipertahankan jika terhubung dengan tepi yang kuat.
7. Hasil tepi citra: Setelah proses *hysteresis thresholding*, hasilnya adalah citra dengan tepi yang telah terdeteksi dengan baik. Tepi kuat akan menjadi bagian dari tepi akhir, sementara tepi lemah yang terhubung dengan tepi kuat juga akan menjadi bagian dari tepi akhir.
8. *End*: Proses deteksi *canny* selesai, dan hasil akhirnya adalah citra dengan tepi yang telah berhasil dideteksi dengan menggunakan metode *canny*.

3.8. Pembelajaran dengan Jaringan Syaraf Tiruan *Backpropagation*

Mengimplementasikan jaringan syaraf tiruan *backpropagation* dengan pelatihan menggunakan dataset gambar Ikan Koi.



Gambar 6. *Flowchart* pembelajaran Jaringan Syaraf Tiruan *Backpropagation*

Berikut adalah penjelasan dari Gambar 6, yaitu pembelajaran Jaringan Syaraf Tiruan Backpropagation**:**

1. Inisialisasi *Epoch* Pelatihan = 1

   Langkah ini menginisialisasi nilai awal untuk variabel *Epoch* yang akan digunakan dalam proses pelatihan. *Epoch* merupakan jumlah iterasi atau putaran dalam proses pelatihan jaringan syaraf.
2. Inisialisasi Bobot dan Bias dengan Nilai Acak

   Pada langkah ini, bobot (*weights*) dan bias diinisialisasi dengan nilai acak. Inisialisasi ini penting karena bobot dan bias akan diperbarui selama proses pelatihan untuk mengoptimalkan kinerja jaringan.
3. Menyajikan Pola Input dan Menghitung Nilai *Output*

   Langkah ini melibatkan penyajian pola masukkan (*input pattern*) ke dalam jaringan dan menghitung nilai *output* yang dihasilkan oleh jaringan. Pada setiap iterasi pelatihan, pola *input* disajikan ke jaringan untuk menghasilkan *output* yang sesuai.
4. Menghitung Galat Kuadrat Rata-rata (MSE)

   Setelah mendapatkan output dari jaringan, langkah ini melibatkan perhitungan *Mean Squared Error* (MSE) atau rata-rata galat kuadrat. MSE mengukur sejauh mana *output* jaringan mendekati nilai target yang diinginkan. Tujuan dalam pelatihan adalah untuk mengurangi MSE sekecil mungkin.
5. *MSE < MSEmin*

   Pada langkah ini, dilakukan pengecekan apakah MSE yang dihasilkan pada iterasi pelatihan

saat ini lebih kecil dari nilai MSE minimal yang ditentukan sebelumnya (MSEmin). Jika MSE lebih kecil dari MSEmin, maka kriteria konvergensi tertentu telah terpenuhi.

6. *Epoch* > *Epochmax*
   Langkah ini melibatkan pengecekan apakah jumlah *Epoch* atau iterasi pelatihan saat ini telah melebihi jumlah *Epoch* maksimum yang ditentukan sebelumnya (*Epochmax*). Jika ya, maka proses pelatihan dihentikan.
7. Memperbarui Bobot dan Bias
   Jika kedua kondisi di atas tidak terpenuhi, maka dilakukan pembaruan bobot (*weights*) dan bias jaringan. Pembaruan ini dilakukan menggunakan algoritma yang sesuai, seperti algoritma *backpropagation*, untuk mengoptimalkan performa jaringan.
8. *Epoch* = *Epoch* + *1*:
   Setelah melakukan pembaruan bobot dan bias, langkah ini menginkrement nilai *Epoch* dengan 1 untuk menandai berakhirnya satu iterasi pelatihan dan mempersiapkan iterasi berikutnya.
9. Menghentikan Pelatihan Jaringan
   Pelatihan jaringan berhenti ketika salah satu kondisi berikut terpenuhi: MSE < MSEmin atau *Epoch* > *Epochmax*. Jika kondisi berhenti terpenuhi, proses pelatihan dihentikan dan jaringan dianggap sudah cukup terlatih.

3.9. Pengujian Jaringan Syaraf Tiruan *Backpropagation*

Pada tahap ini adalah tahapan yang meliputi proses pengujian hasil perhitungan dengan metode *backpropagation* yang telah dilakukan. Data yang telah dikumpulkan dilatih secara bertahap berdasarkan jumlah citra latih untuk mendapatkan jaringan terbaik dan evaluasi hasil pengenalan pola Ikan Koi yang dilakukan oleh jaringan syaraf tiruan *backpropagation*.

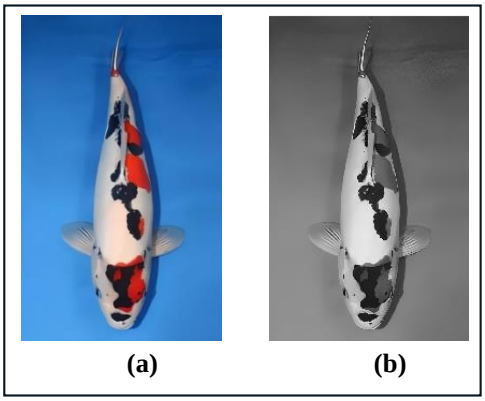
3.10. Analisis Hasil Klasifikasi

Pada tahap ini adalah menganalisa hasil klasifikasi untuk mengevaluasi efisiensi dan akurasi pengidentifikasian pola Ikan Koi dan menarik kesimpulan dari hasil analisa untuk melihat sejauh mana algoritma *canny* dan jaringan syaraf tiruan *backpropagation* berhasil dalam mengidentifikasi Ikan Koi.

4. HASIL DAN PEMBAHASAN

4.1. Proses Deteksi Tepi *Canny*

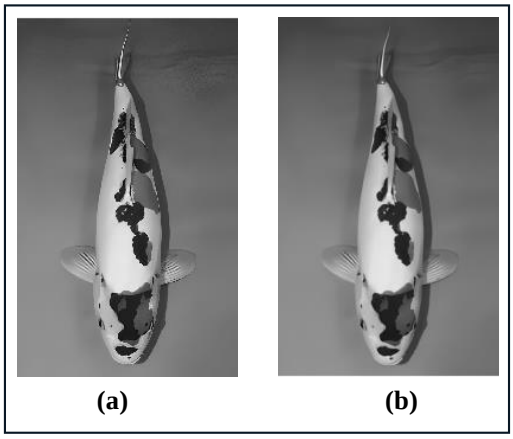
Citra yang digunakan dalam penelitian ini awalnya dibagi menjadi 2 kelompok. Kelompok yang satu sebagai citra latih yang digunakan untuk melatih sistem jaringan yang akan dibuat. Sedangkan kelompok yang lain sebagai citra uji yang digunakan untuk menguji performasi jaringan yang telah dibuat. Dimana masing-masing kelompok terdiri dari tiga jenis citra Ikan Koi. Dalam penelitian ini digunakan 90 citra latih yang terdiri dari 30 Ikan Koi jenis *Sanke*, 30 citra Ikan Koi jenis *Kohaku* dan 30 citra Ikan Koi jenis *Showa*. Sedangkan citra uji yang digunakan adalah 10 citra yang terdiri dari Ikan Koi jenis *Sanke*, 10 citra Ikan Koi jenis *Kohaku* dan 10 citra Ikan Koi jenis *Showa*.



Gambar 7. Citra Asli (a) Citra hasil *grayscale* (b)

Pemrosesan citra awal yang dilakukan adalah pengubahan tipe citra, yang semula bertipe RGB dirubah menjadi tipe *grayscale* atau skala keabuan. Konversi tipe citra ini dimaksudkan untuk mempermudah analisis citra lebih lanjut. Citra *grayscale* memiliki 256 derajat keabuan dengan skala nilai keabuan 0-255. Warna hitam dalam citra ditunjukkan dengan nilai keabuan "0" dan warna putih ditunjukkan dengan nilai "255". Nilai keabuan inilah yang nantinya diolah dalam proses selanjutnya.

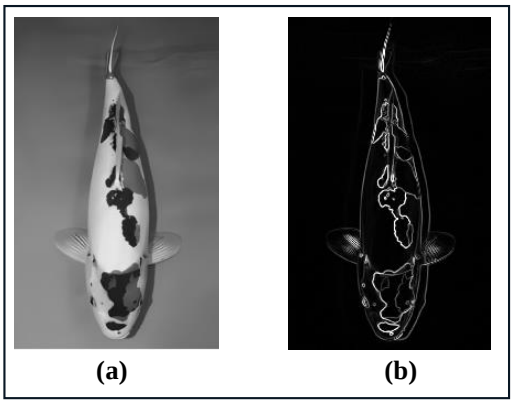
Secara matematis, data yang nantinya akan diolah dan dapat terhitung dalam program adalah nilai keabuan yang termasuk dalam bilangan asli atau nilai keabuan 1 – 255. Dengan kata lain, gambar yang terbaca oleh program *MATLAB* sebagai objek adalah bagian citra dengan warna putih. Proses *grayscaling* terlihat pada Gambar 7.



Gambar 8. Citra hasil *grayscale* (a) Citra hasil filter *Gaussian* (b)

Langkah selanjutnya adalah menerapkan filter *Gaussian* pada citra *grayscale*. Filter ini berfungsi

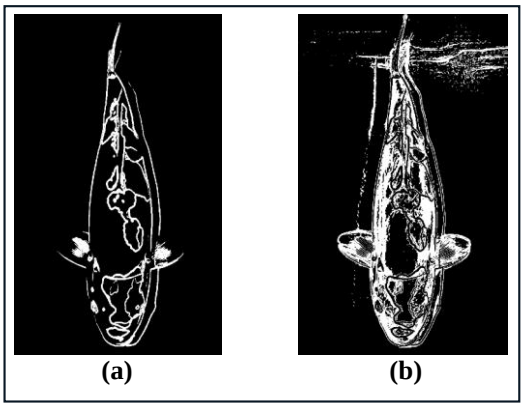
untuk mengurangi *noise* dan memperhalus citra. Dalam proses penghalusan terlebih dahulu kernel dirancang berdasarkan pada ordo matriks m x n dan nilai standar deviasi menggunakan persamaan kemudian dikonvolusi dengan matriks citra. Pada penelitian ini menggunakan matriks dengan ordo 7x7, dikarenakan mencakup piksel-piksel sekitar titik tengah dengan cukup lebar dan menggunakan sigma 2 dalam persamaan *Gaussian* yang berfungsi untuk mengendalikan tingkat penghalusan serta kepekaan terhadap detail dalam citra. Proses filter *Gaussian* terlihat pada Gambar 8.



Gambar 9. Citra hasil filter *Gaussian* (a) Citra hasil *Sobel* (b)

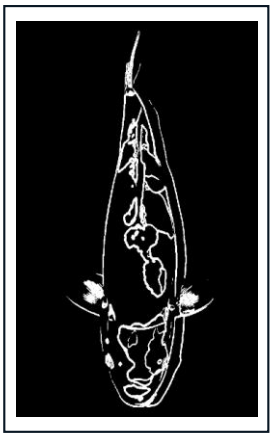
Setelah diterapkan filter *Gaussian*, operator *Sobel* digunakan untuk mendeteksi gradien citra. Operator *Sobel* terdiri dari dua kernel, yaitu kernel *Sobel* horizontal (Gx) dan kernel *Sobel* vertikal (Gy). Kedua kernel ini digunakan untuk menghitung gradien horizontal dan gradien vertikal di setiap piksel citra. Gradien horizontal dan vertikal yang telah dihitung dengan operator *Sobel* digunakan untuk mencari gradien *magnitude* dan arahnya. Proses deteksi dengan operator *Sobel* terlihat pada Gambar 9.

Langkah selanjutnya adalah menerapkan *non-maximum suppression* untuk menghilangkan piksel yang bukan merupakan tepi tajam. Proses ini melibatkan perbandingan piksel gradien dengan piksel sekitarnya dalam arah gradien. Jika piksel gradien merupakan maksimum lokal dalam arah gradien, maka piksel tersebut tetap dipertahankan sebagai tepi, sedangkan piksel yang tidak merupakan maksimum lokal akan dihapus.



Gambar 10. Citra *Strong Edges* (a) Citra *Weak Edges* (b)

Langkah selanjutnya adalah menerapkan *threshoding* ganda (*hysteresis*) untuk mengkategorikan tepi menjadi tepi kuat, tepi lemah, atau bukan tepi. Tepi kuat adalah piksel yang memiliki gradien yang cukup besar dan jelas merupakan tepi, sedangkan tepi lemah adalah piksel yang memiliki gradien lebih rendah dan mungkin merupakan tepi atau *noise*. Piksel bukan tepi adalah piksel yang memiliki gradien yang rendah dan dianggap bukan tepi. *Threshoding* ganda menggunakan batas atas (*upper threshold*) dan batas bawah (*lower threshold*) untuk mengkategorikan piksel menjadi tepi kuat (*strong edges)*, tepi lemah (*weak edges*), atau bukan tepi. Dalam penelitian ini menggunakan nilai batas atas 45 dan nilai batas bawah 5, tepi dengan *magnitude* gradien di bawah 5 akan diabaikan, sedangkan tepi dengan *magnitude* gradien di atas 45 akan dianggap sebagai tepi yang kuat. Tepi dengan magnitude gradien antara 5 dan 45 akan dianggap sebagai tepi yang lemah, tetapi tetap dipertahankan jika terhubung dengan tepi yang kuat. Proses *threshoding* ganda (*hysteresis thresholdong*) terlihat pada Gambar 10.



Gambar 11. Citra hasil *Canny*

Hasilnya adalah citra dengan tepi yang telah terdeteksi dengan baik. Terlihat pada Gambar 11 tepi kuat akan menjadi bagian dari tepi akhir, sementara tepi lemah yang terhubung dengan tepi kuat juga akan menjadi bagian dari tepi akhir.

4.2. Ekstraksi Fitur Citra

Sebelum citra dapat diidentifikasi, diperlukan informasi karakter dari citra yang dapat menjadi pembanding antara citra Ikan Koi dengan jenis yang lainnya yang lebih dikenal dengan fitur. Fitur yang digunakan adalah nilai rata-rata (*mean*) untuk pola warna, nilai imbinarize untuk pola bentuk, dan nilai *graycomatrix* (GLCM) untuk tekstur. Berikut contoh ekstraksi fitur dari salah satu citra latih Ikan Koi:

mean = 0.400000000000000
imbinarize = -0.200000000000000
graycomatrix = 0.0450635223155795

Fungsi *mean* adalah untuk menghitung nilai rata-rata dari suatu data. Pada konteks ekstraksi fitur, nilai rata-rata sering digunakan sebagai representasi fitur tertentu, seperti rata-rata intensitas piksel atau rata-rata suatu distribusi.

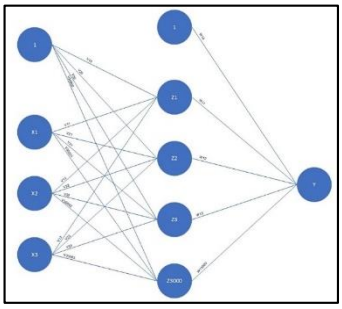
Fungsi *imbinarize* adalah untuk mengubah citra menjadi citra biner, di mana setiap piksel memiliki nilai 0 (hitam) atau 1 (putih) tergantung pada ambang batas yang ditentukan. Hal ini berguna dalam mengkonversi citra menjadi representasi biner.

Fungsi *graycomatrix* adalah untuk menghitung matriks ko-eksisensi level abu-abu (GLCM) dari citra. GLCM adalah representasi statistik yang menggambarkan hubungan antara piksel dengan nilai abu-abu tertentu pada jarak dan arah tertentu dalam citra. GLCM digunakan untuk mengukur berbagai fitur tekstur seperti kontras, korelasi, energi, dan homogenitas.

4.3. Pelatihan Jaringan Syaraf Tiruan *Backpropagation*

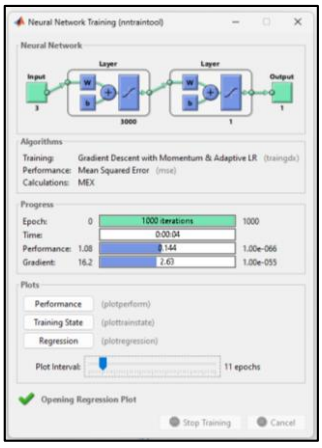
Identifikasi citra Ikan Koi dalam penelitian ini menggunakan metode jaringan syaraf tiruan *backpropagation* dimana di dalamnya terdapat dua proses, yakni proses pelatihan jaringan dan proses pengujian jaringan. Proses pelatihan digunakan untuk mencari bobot dan proses pengujian digunakan untuk menguji sistem dengan menggunakan bobot yang telah didapatkan pada proses pelatihan.

Pada proses pelatihan ini dilakukan dengan menggunakan data 90 citra latih yang telah dipilih sebelumnya. Dalam proses ekstraksi fitur sebelumnya data ini telah disimpan dalam bentuk *database*. Sehingga ketika proses pelatihan dijalankan maka data dapat dimasukkan secara otomatis.



Gambar 12. Arsitektur JST *Backpropagation* yang digunakan dalam penelitian

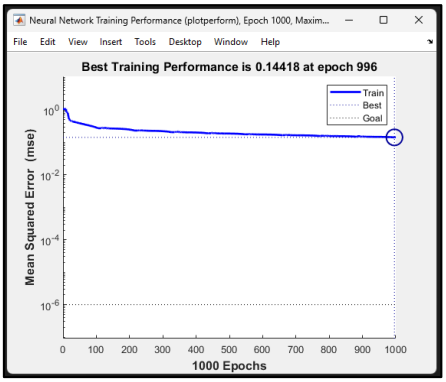
Pada Gambar 12 jaringan syaraf tiruan *backpropagation* memiliki arsitektur jaringan yang terdiri dari *input layer*, *hidden layer* dan *output layer*. Pembangunan sistem jaringan dilakukan dengan mengubah parameter-parameter yang ada di dalamnya, seperti jumlah *layer* tersembunyi, jumlah *neuron* pada *layer* tersembunyi, dan fungsi aktivasi. Pada penelitian ini, jaringan yang dibentuk memiliki satu *layer* tersembunyi (*hidden layer*) dimana *layer* tersembunyi mengandung 3.000 *neuron*, *max epoch* atau jumlah iterasi yang dilakukan selama proses pelatihan adalah 1.000 (peatihan akan berjalan hingga mencapai 1.000 *epoch* atau jika kondisi berhenti lainnya terpenuhi), *learning rate* atau laju proses pelatihan yang digunakan adalah 0.1 (yang menunjukkan penyesuaian bobot yang relatif besar pada setiap langkah pelatihan), dan *max error* atau nilai *error* yang dihasilkan setelah proses pelatihan selesai (pelatihan akan berhenti jika kesalahan pelatihan mencapai atau kurang dari 1e-6. Hal ini menandakan bahwa jaringan mencapai tingkat akurasi yang diinginkan ketika kesalahan pelatihan sangat kecil).



Gambar 13. Hasil pelatihan JST *Backpropagation* menggunakan *toolbox nntraintool*

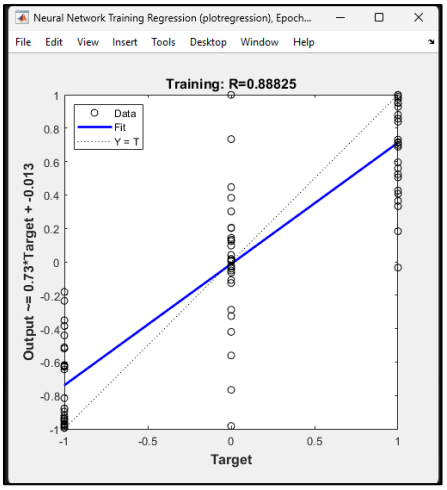
Proses pelatihan menggunakan *toolbox nntraintool* yang telah disediakan oleh *MATLAB*. Informasi yang diberikan dari hasil pelatihan yaitu *max epoch*, jumlah *hidden layer*, *learning rate*, dan

max error. Proses pelatihan menggunakan traintool dapat dilihat pada Gambar 13.



Gambar 14. Performa pelatihan JST *Backpropagation* menggunakan *toolbox nntraintool*

Dalam proses pelatihan jaringan syaraf tiruan, terdapat pernyataan bahwa kinerja terbaik telah dicapai pada iterasi ke-996 dengan nilai performa sebesar 0.14418. Hal ini menunjukkan bahwa pada titik tersebut, jaringan mencapai hasil yang paling baik dalam mempelajari pola dari data yang diberikan. Performa pelatihan jaringan syaraf tiruan *backpropagation* menggunakan *toolbox nntraintool* dapat dilihat pada Gambar 14.



Gambar 15. Regresi pelatihan JST *Backpropagation* menggunakan *toolbox nntraintool*

Dalam pelatihan jaringan syaraf tiruan menggunakan *nntraintool* di *MATLAB*, "*Training*=0.88825" menunjukkan bahwa kinerja pelatihan jaringan tersebut mencapai nilai 0.88825. Nilai ini merupakan ukuran performa jaringan berdasarkan metrik tertentu, seperti *mean squared error* (MSE) atau *R-squared*. Semakin rendah nilai training, semakin baik kinerja jaringan dalam menghasilkan prediksi yang akurat.

Selain itu, "*Output*~=0.73*Target+-0.013" menggambarkan hubungan antara *output* yang diprediksi oleh jaringan (*Output*) dengan nilai target yang sebenarnya (Target) menggunakan persamaan linier. Persamaan ini menunjukkan bahwa *output* yang diprediksi secara kasar dapat diperoleh dengan mengalikan nilai target dengan faktor 0.73 dan menambahkan *offset* -0.013.

Grafik *training regression* digunakan untuk memvisualisasikan perubahan kinerja jaringan selama proses pelatihan. Sumbu x pada grafik menunjukkan iterasi pelatihan atau *epoch*, sementara sumbu y menunjukkan performa atau kesalahan prediksi jaringan. Tujuannya adalah untuk memantau penurunan kesalahan prediksi dan peningkatan performa jaringan seiring dengan berjalannya iterasi pelatihan. Jika grafik menunjukkan bahwa kesalahan prediksi telah stabil dan performa jaringan telah mencapai tingkat yang diinginkan, maka pelatihan dapat dihentikan dan jaringan dapat digunakan untuk melakukan prediksi pada data baru. Regresi pelatihan jaringan syaraf tiruan *backpropagation* menggunakan *toolbox nntraintool* dapat dilihat pada Gambar 15.

Tabel 1. Pengujian Citra Latih

Jumlah Citra Latih	Jumlah Dikenali	Learning Rate	MSE	Prosentase Keberhasilan
45	38	0,1	0,87192	84,4444%
45	37	0,2	0,8896	82,2222%
45	36	0,3	0,8745	80,0000%
60	50	0,1	0,88911	83,3333%
60	49	0,2	0,88432	81,6667%
60	48	0,3	0,89367	80,0000%
90	74	0,1	0,88825	82,2222%
90	71	0,2	0,88919	78,8889%
90	70	0,3	0,86269	77,7778%

Pada Tabel 1 dapat dilihat hasil dari pengujian citra latih yang dilakukan, berikut adalah penjelasannya:

1) Data pertama, terdapat 45 citra latih dengan 38 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.1. Dalam pelatihan, diperoleh nilai MSE sebesar 0.87192, dan persentase keberhasilan mencapai 84.4444%.
2) Data kedua menunjukkan hasil pelatihan dengan 45 citra latih dan 37 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.2. Dalam pelatihan, diperoleh nilai MSE sebesar 0.8896, dan persentase keberhasilan sebesar 82.2222%.
3) Data ketiga, terdapat 45 citra latih dengan 36 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.3. Hasil pelatihan menunjukkan nilai MSE sebesar 0.8745, dan persentase keberhasilan mencapai 80.0000%.
4) Data keempat menampilkan hasil pelatihan dengan 60 citra latih dan 50 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.1. Dalam pelatihan, diperoleh nilai MSE sebesar 0.88911, dan persentase keberhasilan mencapai 83.3333%.

5) Data kelima, terdapat 60 citra latih dengan 49 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.2. Dalam pelatihan, diperoleh nilai MSE sebesar 0.88432, dan persentase keberhasilan sebesar 81.6667%.
6) Data keenam menunjukkan hasil pelatihan dengan 60 citra latih dan 48 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.3. Dalam pelatihan, diperoleh nilai MSE sebesar 0.89367, dan persentase keberhasilan mencapai 80.0000%.
7) Data ketujuh, terdapat 90 citra latih dengan 74 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.1. Hasil pelatihan menunjukkan nilai MSE sebesar 0.88825, dan persentase keberhasilan mencapai 82.2222%.
8) Data kedelapan menampilkan hasil pelatihan dengan 90 citra latih dan 71 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.2. Dalam pelatihan, diperoleh nilai MSE sebesar 0.88919, dan persentase keberhasilan sebesar 78.8889%.
9) Data kesembilan, terdapat 90 citra latih dengan 70 citra yang berhasil dikenali. *Learning rate* yang digunakan adalah 0.3. Hasil pelatihan menunjukkan nilai MSE sebesar 0.86269, dan persentase keberhasilan mencapai 77.7778%.

4.4. Pengujian Jaringan Syaraf Tiruan *Backpropagation*

Data yang telah dikumpulkan dilatih secara bertahap berdasarkan penambahan jumah citra latih untuk mendapatkan jaringan terbaik. Pada tahap pertama digunakan citra latih sebanyak 45 citra, tahap kedua digunakan citra latih sebanyak 60, dan tahap ketiga digunakan citra latih sebanyak 90 citra. Pada proses pelatihan ada 2 parameter yang digunakan sebagai acuan pelatihan yaitu *learning rate* dan MSE seperti yang terlihat pada tabel berikut:

Tabel 2. Pengujian Citra Uji

Jumlah Citra Latih	Jumlah Citra Uji	Jumlah Dikenali	Prosentase Keberhasilan
45	30	19	63,3333%
60	30	20	66,6667%
90	30	21	70,0000%

Pada Tabel 2 dapat dilihat hasil dari pengujian citra uji terhadap citra latih, berikut adalah penjelasannya:

1) Pada data pertama, terdapat 45 citra latih dan 30 citra uji. Dalam pengujian, jaringan berhasil mengenali 19 citra dengan tingkat keberhasilan sebesar 63,3333%.
2) Data kedua menunjukkan hasil pengujian dengan 60 citra latih dan 30 citra uji. Jaringan mampu mengenali 20 citra dengan tingkat keberhasilan sebesar 66,6667%.
3) Pada data ketiga, terdapat 90 citra latih dan 30 citra uji. Hasil pengujian menunjukkan bahwa jaringan mampu mengenali 21 citra dengan tingkat keberhasilan sebesar 70,0000%.

4.5. Pembahasan

Dalam penelitian ini, terdapat 3 data yang menunjukkan hasil pengenalan citra menggunakan jaringan. Data pertama menunjukkan bahwa dengan 45 citra latih dan 30 citra uji, jaringan mampu mengenali 19 citra dengan tingkat keberhasilan 63,3333%. Data kedua menunjukkan hasil yang sedikit lebih baik, dengan 60 citra latih dan 30 citra uji, jaringan mampu mengenali 20 citra dengan tingkat keberhasilan 66,6667%. Pada data ketiga, dengan 90 citra latih dan 30 citra uji, jaringan mampu mengenali 21 citra dengan tingkat keberhasilan 70,0000%.

5. KESIMPULAN DAN SARAN

Kesimpulan dari penelitian ini adalah bahwa Algoritma *Canny* efektif dalam mengidentifikasi Ikan Koi jenis *Sanke*, *Kohaku*, dan *Showa* dengan tingkat presisi yang tinggi, sementara penggunaan metode Jaringan Syaraf Tiruan *Backpropagation* berhasil dalam pengenalan Ikan Koi pada tahap pelatihan dan pengujian, terutama dengan *learning rate* 0.1. Kombinasi kedua metode ini melibatkan pengenalan tepi dan fitur gambar Ikan Koi melalui Algoritma *Canny*, yang kemudian digunakan sebagai input dalam Jaringan Syaraf Tiruan *Backpropagation*. Untuk meningkatkan performa sistem, disarankan untuk meningkatkan jumlah data pelatihan, melakukan analisis fitur yang relevan, dan membangun jaringan syaraf tiruan yang sesuai dengan kebutuhan, agar hasil pengenalan Ikan Koi dapat ditingkatkan.

DAFTAR PUSTAKA

[1] M. Yetri Y, "IDENTIFIKASI POLA WARNA IKAN KOI MENGGUNAKAN METODE SOBEL EDGE DETECTION DALAM KARAKTERISTIK CITRA SHARPENING," Jurnal SAINTIKOM, vol. 14, 2015, pp. 45-48.
[2] N. Hermana and M. S. J. Juerman, "Implementasi Algoritma Canny dan Backpropagation dalam Pengenalan Pola Rumah Adat," Jurnal Itenas, 2014.
[3] R. D. Atmaja, "DETEKSI JENIS KAYU CITRA FURNITURE UKIRAN JEPARA," in Konferensi Nasional Sistem Informasi, STIKOM Bali, 2012.
[4] E. W., "Aplikasi Deteksi Tepi pada Realtime Video," Jurnal Teknologi Informasi DINAMIK, vol. 16, 2011, pp. 44-49.
[5] D. Kusumaningsih and S. A. P. Pramudita, "PENGGUNAAN METODE BACKPROPAGATION ARTIFICIAL NEURAL NETWORK DALAM SISTEM PENGENALAN NOTASI BALOK MENJADI MIDI," JURNAL TELEMATIKA MKOM, 2016, vol. 8.
[6] Haris and A. Prasetyo, "IMPLEMENTASI METODE DETEKSI TEPI CANNY PADA

OBJEK SEBAGAI MODEL KEAMANAN APLIKASI PADA SMARTPHONE ANDROID," JURNAL PETIR, 2016, pp. 1-87.

[7] Wicaksono, "ANALISIS IDENTIFIKASI POLA IKAN KOI MENGGUNAKAN METODE SOBEL EDGE DETECTION," Simki-Techsain, vol. 1, 2017.

[8] V. Pebrianasari, "ANALISIS PENGENALAN MOTIF BATIK PEKALONGAN MENGGUNAKAN ALGORITMA BACKPROPAGATION," Techno.COM, vol. 14, 2015, pp. 281-290.

[9] Rozani, "PENERAPAN METODE JARINGAN SYARAF TIRUAN PADA APLIKASI PENGENALAN BAHASA ISYARAT ABJAD JARI," JATI (Jurnal Mahasiswa Teknik Informatika), 2017.

[10] D. R. Kishore and T. Kaur, "Backpropagation Algorithm: An Artificial Neural Network Approach for Pattern Recognition," International Journal of Scientific & Engineering Research, vol. 3, no. 6, 2012.