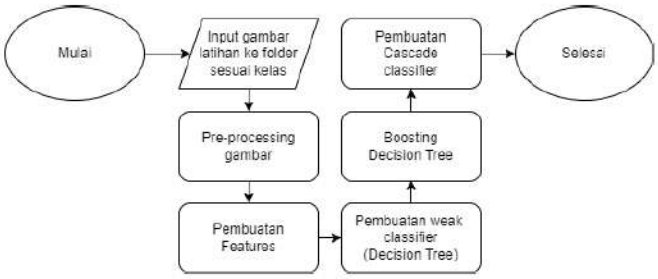
BAB III

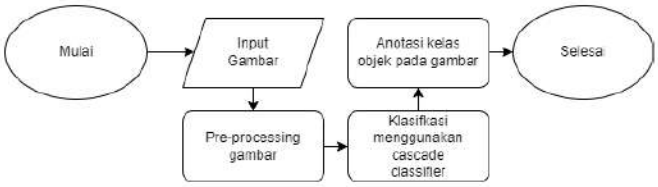
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Gambar *flowchart* berikut mengilustrasikan proses pelatihan dari dataset dan juga proses penggunaan yang sesungguhnya.



Gambar 3.1: Diagram alir untuk algoritma pelatihan klasifikasi objek



Gambar 3.2: Diagram alir untuk algoritma klasifikasi objek

3.2 Desain Sistem

Dalam proses pembuatan *classifier* perlu dilewati tahapan *training*. Tujuan *training* adalah untuk menciptakan suatu *strong classifier* yang nantinya dapat digunakan untuk melakukan klasifikasi yang sebenarnya. Pertama, gambar yang akan menjadi contoh pelatihan dianotasi sesuai kelasnya masing-masing dengan memberikan label yang sesuai dengan kelas mereka masing-masing, contoh latihan ini berisikan gambar-gambar yang tidak memiliki kelas yang benar, atau *false example* dan juga gambar-gambar yang memiliki kelas yang benar, atau *positive example*. Setelah itu gambar melalui proses *pre-processing* dan disesuaikan untuk mengoptimalkan proses pelatihan.

Pertama sebuah set *features* akan dibuat dengan cara mencoba semua kemugnkinan yang ada dengan bentuk fitur yang dimiliki. Set ini akan berisikan

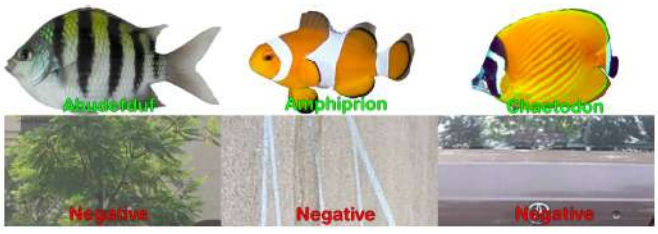
informasi fitur-fitur yang nantinya akan dipakai untuk mendapatkan nilai fitur yang sesungguhnya. Set gambar latihan lalu akan dibaca menggunakan semua fitur ini dan hasilnya akan dicatat dalam tabel csv. Algoritma lalu akan mengkonstruksi sebuah *decision tree* untuk setiap *feature* untuk menentukan nilai *feature threshold* setiap kelas. *Decision tree* lalu akan di-*boosting* untuk menentukan nilai bobot votingnya pada *strong clossifier*. Akhrinya dari sekumpulan *decision tree* ini dibuatlah sebuah *final strong classifier* yang berbentuk *cascade*.

Dengan *final strong classifier*, barulah klasifikasi objek yang sesungguhnya dapat dilakukan. Pertama gambar yang akan diklasifikasi akan melalui *pre-processing*. setiap sub-window akan dicek menggunakan *strong classifier* untuk menentukan kelasnya. Hal ini dilakukan pada tiga area: area kiri untuk mengklasifikasi mulut dari ikan, area tengah untuk mengklasifikasi sirip dari ikan, dan terakhir area kanan untuk mengklasifikasi bentuk ekor dari ikan. Hasil ketiga *sub-window* ini nantinya juga akan ber-*voting* dimana jika ada dua atau lebih *sub window* berhasil mengklasifikasikan kelas yang sama, maka kelas itu dipilih sebagai kelas dari objek pada gambar. Kelas dari objek lalu akan dituliskan di pojok kiri atas gambar.

3.3 Training *Strong Classifier*

Pada *Training* ada tiga tahapan yang perlu dijalankan untuk menghasilkan sebuah *Strong Classifier*, yaitu penginputan *dataset* pelatihan yang sudah dianotasi, pembuatan *features*, pembuatan *decision tree* untuk setiap *features*, *Boosting* dan pemilihan fitur untuk pemebuatan *attentional cascade*.

3.3.1 *Input Dataset* Pelatihan

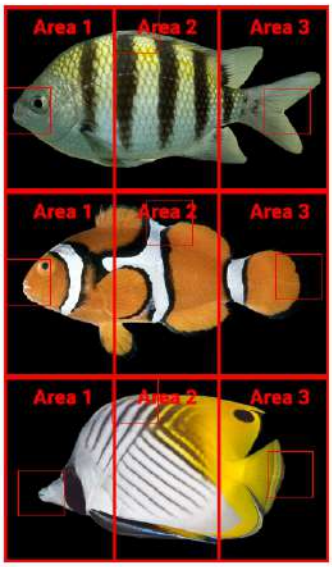


Gambar 3.3: Contoh gambar Abudefduf, Amphiprion, Chaetodon dan contoh gambar-gambar negatif

Dataset yang akan dipakai diambil dari FishBase (Dapat dilihat di *FishBase* 2024: https://fishbase.mnhn.fr/) yang berisikan berbagai gambar ikan termasuk gambar dari genus Abudefduf, gambar dari genus Amphiprion dan gambar dari genus Chaetodon. Ketiga genus ikan ini dipilih karena bentuknya yang berbeda satu-sama lain. Untuk contoh pelatihan gambar dibuat berukuran 350x200 piksel, dengan warna *greyscale*. Selain itu juga akan dipilih contoh pelatihan negatif atau kumpulan gambar yang tidak terdapat kelas ikan dari http://www.vision.caltech.edu/datasets/ dengan jumlah yang sama, resolusi sama dan perlakuan yang sama. Anotasi dilakukan dengan menyimpan label dalam sebuah *array*, label diambil dari sumber folder gambar. Kelas 0 direservasi untuk kelas negatif, sementara kelas 1, 2 dan 3 direservasi untuk Abudefduf, Amphiprion dan Chaetodon. *Dataset* ini lalu dibagi menjadi tiga yaitu *training dataset*, *testing dataset*, dan *validation dataset* dengan jumlah yang sama.

Semua gambar akan di-*load* menggunakan *library* CV2. *Library* ini adalah sebuah *library* yang dapat digunakan untuk membaca gambar dan mengubahnya menjadi sebuah bentuk *array 2d*, dimana *array* akan memiliki ukuran persis dengan ukuran gambar. Selain itu CV2 juga akan digunakan untuk melakukan anotasi pada akhir klasifikasi.

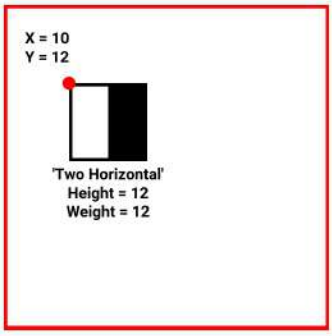
Selain itu untuk setiap kelas sudah dibuatkan *sub-window* spesifik untuk diklasifikasi. *Sub-window* ini secara spesifik mengklasifikasi tiga bagian ikan yaitu, mulut, sirip dan ekor. klasifikasi spesifik ini nantinya akan bekerja-sama dengan *sliding window* dalam mengklasfikasi gambar yang ada, dengan mencari mulut, sirip dan ekor ikan didalam gambar.



Gambar 3.4: *sub-window* setiap kelas ikan yang akan dipelajari oleh *classifier*

3.3.2 Pembuatan *Haar like Features*

Fitur-fitur yang akan digunakan dalam klasifikasi dibuat berdasarkan *Haar-like Features* dan berisikan informasi penting yang dapat digunakan untuk mencari nilai sebuah fitur. Sebuah fitur berisikan tipe fiturnya, lokasi fitur tersebut didalam *sub-window*, dan ukuran dari fitur tersebut. Misalnya ada sebuah fitur, ia bertipe dua persegi panjang menghadap ke kiri, lokasi x-nya adalah 12 piksel, dan lokasi y-nya adalah 10 piksel, dia memiliki ukuran 8 x 8 piksel.



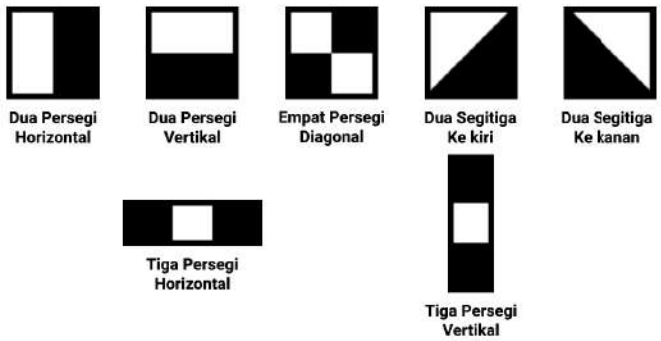
Gambar 3.5: Sebuah fitur dua persegi panjang menghadap ke kiri, lokasi x = 12 piksel, lokasi y = 10 piksel, dengan ukuran 12 x 12 piksel.

Dengan informasi yang ada didalam sebuah fitur tersebut, nilai fitur dapat dicari dengan mudah menggunakan rumus:

$$\sum \text{nilai piksel area putih} - \sum \text{nilai piksel area hitam} \tag{3.1}$$

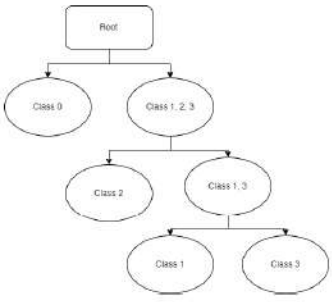
Pada tahap ini fitur yang dipilih akan dibuat untuk semua posibilitas lokasi yang ada dan ukuran yang ada. Hal ini dilakukan dengan membuat fitur dimulai dari kiri atas gambar dengan ukuran 2 x 2 piksel untuk fitur dua persegi panjang, empat persegi dan diagonal. Dan fitur 1 x 3 piksel atau 3 x 1 piksel untuk fitur tiga persegi. Hal ini juga dilakukan sampai ukuran fitur lebih besar dari *sub-window* dan tidak bisa muat lagi.

Beberapa jenis *feature* yang digunakan adalah 2 persegi horizontal, 2 persegi vertikal, 4 persegi diagonal, 2 segitiga hadap kiri, 2 segitiga hadap kanan, 3 persegi horizontal, dan 3 persegi vertikal.



Gambar 3.6: Gambaran fitur-fitur yang akan digunakan

3.3.3 Pembuatan *Decision Tree*



Gambar 3.7: Contoh sebuah *decision tree* dengan kelas 0, 1, 2 dan 3

Setiap *weak learner* adalah sebuah *decision tree* yang akan mengambil nilai dengan fitur untuk digunakan sebagai variabel klasifikasi. fitur dapat digunakan untuk

mencari sebuah nilai perbandingan intensitas cahaya dari dua area pada gambar dan mendeteksi keberadaan suatu fitur seperti perbedaan warna, garis, maupun perbedaan kontras pada gambar.

Decision tree pertama akan dibuat dari *root* atau akar, yang lalu akan bercabang hingga batas maksimum telah dicapai. *threshold* yang digunakan dalam pembuatan *decision tree* adalah salah satu nilai fitur yang dibaca dari gambar. Dengan cara ini, *decision tree* tidak perlu mencoba semua nilai yang mungkin, dan dengan demikian mempercepat proses pembuatan *decision tree*. Batas maksimum tinggi *decision tree* yang dipilih adalah tiga tingkat, hal ini dikarenakan waktu komputasi yang memakan waktu bila *decision tree* memiliki lebih dari 3 tingkat. Di lain sisi, menggunakan tingkat kurang dari tiga tidak mungkin memenuhi persyaratan klasifikasi empat kelas.

3.3.4 *Boosting*

Boosting ditunjukan untuk memberikan nilai *voting* untuk setiap *weak classifier* yang nantinya akan digunakan dalam klasifikasi akhir. Sebelum *boosting* dimulai, *weak classifier* yang kurang diskriminatif akan dibuang. Hal ini dilakukan dengan membandingkan hasil prediksi *weak classifier* dengan label yang sebenarnya, dimana *weak classifier* yang gagal memprediksi 50% dari set tes akan dibuang. Ini dilakukan untuk mengurangi jumlah *weak classifier* yang akan dipakai berikutnya.

Pada tahap ini *boosting* akan dijalankan dari *decision tree* yang paling akurat ke yang paling tidak akurat. Penentuan akurasi ini dilakukan dengan membandingkan label contoh validasi dengan hasil prediksi setiap *weak classifier*. *Weak classifier* yang tidak akurat akan mendapat suara *voting* yang lemah. Rumus untuk mencari bobot voting α dari sebuah *weak classifier* adalah sebagai berikut:

$$\begin{aligned} \epsilon &= \frac{\sum_i \text{image weights}_i \times \text{indikator}_i}{\sum_i \text{image weights}_i} \\ \alpha &= 0.5 \times \log\left(\frac{1-\epsilon}{\epsilon + 1e - 10}\right) \end{aligned} \tag{3.2}$$

weak classifier terbaik akan mulai dan mengklasifikasi seluruh contoh *dataset* validasi dan mencatat contoh yang gagal diklasifikasi oleh *weak classifier* tersebut. Lalu bobot dari contoh tersebut akan dinaikan, dengan maksud agar bobot *voting*

fitur yang paling akurat akan lebih tinggi daripada fitur-fitur yang hanya mendekati menebak. Rumus menaikan bobot dan normalisasi bobot:

$$\text{image weights} \times = \exp(\alpha \times \text{indikator}) \tag{3.3}$$

$$\text{image weights} \div = \sum \text{image weights} \tag{3.4}$$

weak classifier berikutnya lalu akan melakukan prediksi dengan set yang sama, namun dengan bobot gambar yang sudah berubah karena *weak classifier*. Sementara bobot *voting weak classifier* sebelumnya akan disimpan ke array untuk digunakan nanti.

Ketika semua *weak learner* sudah dicari bobot *voting*-nya, akan dibandingkan akurasi *strong cloassifier* yang dibuat dengan iterasi sebelumnya. Bila didapat bahwa ada penurunan akurasi, maupun tidak ada perubahan, maka iterasi Boosting akan disudahi dan *strong cloassifier* pada iterasi ini menjadi *final strong classifier*.

3.3.5 Pembuatan *Attentional Cascade*

Karena bobot *voting* dan urutan *weak classifier* sudah ditentukan pada tahap *Boosting*, pada tahap ini hanya perlu membagi *weak classifier* menjadi beberapa *stage* yang nantinya bisa dipanggil secara terpisah. Sebuah *stage* berlaku layaknya sebuah *strong cloassifier* kecil yang ditargetkan hanya memiliki tingkat akurasi paling tidak 50% saja. Hal ini dilakukan agar klasfikasi seluruh *sub-window* dapat dilakukan tanpa harus memanggil keseluruhan dari *strong classifier*. Metode konstruksi sebuah *stage cascade* adalah sebagai berikut:

Algorithm 3 Cascade Train Stage

```
1:  function TRAIN_STAGE
2:      detection_rate ← 0
3:      while detection_rate < 0.5 do
4:          if len(features) == 0 then
5:              break
6:          end if
7:          self.features.append(features)
8:          detection_rate ← accuracy_score()
9:      end while
10: end function
```

Nantinya dengan memanggil keseluruhan *cascade* untuk melakukan klasifikasi, *stage* dengan satu persatu mengklasifikasi *sub-window* yang sedah dicek. Bila sebuah *stage* mem-*voting* sebuah *sub-window* sebagai kelas negatif (Dalam hal ini voting seluruh *weak classifier* didalam *cascade* mengembalikan kelas 0 atau negatif) maka *stage* akan menghentikan klasifikasi untuk *sub-window* tersebut bila mayoritas *voting* sampai *stage* tersebut memilih negatif atau 0 dan melanjutkan ke *sub-window* berikutnya. Sebaliknya bila *stage* mengembalikan kelas selain 0 (1, 2, maupun 3) maka klasifikasi akan dilanjutkan untuk *sub-window* tersebut sampai salah satu *stage* lainnya mem-*voting* kelas negatif atau semua *stage* habis. Dalam situasi habis, *voting* kelas akan mengambil suara dari semua *weak classifer* dari semua *stage* untuk menentukan *class* pada *sub-window*.

3.4 Skenario Eksperimen dan Validasi

Tahapan ini adalah penggunakaan *classifier* yang sebenarnya dengan tujuan memvalidasi akurasi dari *classifier* tersebut. Gambar ikan yang akan dipakai dalam proses validasi akan melalui beberapa langkah dalam tahap ini, yaitu: *Pre-processing* dalam bentuk *grayscaling*, dan klasifikasi yang sesungguhnya menggunakan *cascade* yang telah dibuat dengan metode *sliding window*. Terlebih, klasifikasi akan dilakukan di tiga area berbeda pada gambar: area kiri untuk mengklasifikasi mulut, area tengah untuk mengklasifikasi sirip, dan area kanan untuk mengklasifikasi ekor. Pada akhirnya ketiga *cascade* dari ketiga area tersebut akan mem-*voting* kelas dari objek pada gambar. Bila didapat bahwa ketiga *cascade* tersebut mem-*voting* tiga kelas yang berbeda, maka kelas yang dipilih adalah 0 atau negatif.

3.4.1 *Pre-processing*

Untuk klasifikasi sebenarnya, gambar yang akan digunakan akan diproses terlebih dahulu. Gambar awalnya akan melakui proses *pre-processing* dan dirubah ke dalam warna *grayscale*. Selain itu resolusi gambar akan dirubah menjadi 350 x 200 piksel agar sesuai dengan ukuran gambar latihan.

3.4.2 Klasifikasi

Pada tahap ini *sliding window* akan mulai bergerak dari pojok kiri atas untuk memulai klasifikasi pada area kiri menggunakan *cascade* yang sudah dilatih untuk mengklasifikasi mulut ikan. Hal ini dilakukan hingga seluruh *sub-window* sudah diklasifikasi dengan hasil negatif, atau bila salah satu *sub-window* mengklasfikasi salah satu kelas positif. Setelah itu maka klasifikasi pada area tengah atau sirip akan dimulai, dan setelahnya baru area kanan atau ekor.

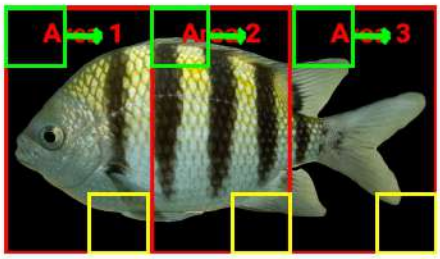
cascade bekerja agak berbeda pada klasifikasi sebenarnya. Melainkan daripada menhitung nilai intensitas cahaya dan fitur pada semua piksel pada gambar, *cascade* hanya menghitung nilai fitur yang berhubungan dengan *weak classifier* dalam *stage cascade* sedang melakukan klasifikasi.

Algorithm 4 Final Cascade Classification

```
function FINALCASCADECLASSIFICATION(image, x_offset, y_offset)
    scoreboard ← array([0, 0, 0, 0])
    for i in range(len(self.stages)) do
        stage_scoreboard ← self.stages[i].stage_prediction()
        if stage_scoreboard.argmax() == 0 then break
        else
            scoreboard ← scoreboard + stage_scoreboard
        end if
    end for
    return scoreboard.argmax()
end function
```

Algorithm 5 Stage Prediction

```
function STAGEPREDICTION(image, x_offset, y_offset, scoreboard)
    for i in range(len(self.features)) do
        feature_type, x, y, width, height ← self.features[i]
        x ← x + x_offset
        y ← y + y_offset
        updated_feature ← (feature_type, x, y, width, height)
        data_features ← compute_feature_with_matrix(image, 0, updated_feature)
        prediction ← self.trees[i].predict(data_features)
        scoreboard[prediction] ← scoreboard[prediction] + 1 × self.alpha_list[i]
    end for
    return scoreboard
end function
```



Gambar 3.8: titik awal *sliding window* (kotak hijau) dan titik akhir (kotak kuning)

3.4.3 Anotasi

Gambar 3.9: Anotasi gambar ikan yang sudah diklasifikasi

Setelah ketiga *window* sudah diklasifikasi oleh *cascade classifer* menggunakan *sliding window*, masing-masing hasilnya akan dipakai untuk menentukan kelas dari objek pada gambar. Setelahnya nama genus ikan akan ditulis pada kiri atas gambar pada gambar hasil.