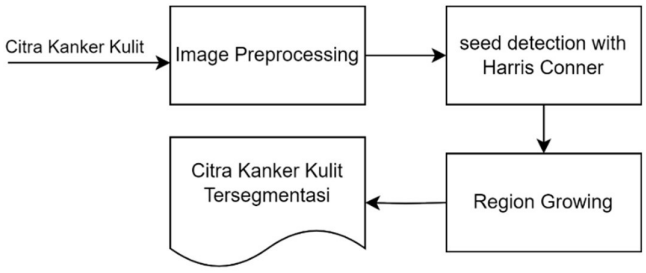
BAB IV
Hasil Dan Pembahasan

4.1 Hasil Segmentasi Metode Region Growing

Sebelum melakukan eksperimen deteksi penyakit kanker kulit dengan metode *LSTM*, akan dilakukan serangkaian eksperimen untuk menguji efektivitas dan keandalan algoritma *region growing* dalam segmentasi citra yang akan digunakan hasil segmentasinya untuk training data *LSTM.*

Metode *Region Growing* adalah metode segmentasi berdasarkan *seed point* . Untuk memulai proses segmentasi dengan metode tersebut diperlukan penentuan *seed point* terlebih dahulu. Untuk mendapatkan *seed point* pada penelitian ini akan menggunakan metode *haris connering detection*. *Seed point* akan ditetapkan berdasarkan sudut dengan nilai intensitas tertinggi dari hasil deteksi sudut dengan *haris connering detection.* Dengan telah adanya *seed point* maka segmentasi dengan *region growing* dapat dilakukan. Untuk gambaran proses eksperimen segmentasi *Region Growing* secara keseluruhan dapat dilihat pada gambar 4.1.



Gambar 4.1. Flowchart Eksperimen Region Growing

Tahap pertama pada eksperimen segmentasi region growing adalah menguji pengaruh ukuran citra dan *neighbors* terhadap akurasi segmentasi. Dari hasil eksperimen didapat Untuk ukuran citra 256x256 dan *neighbors* 4x4, diperoleh hasil akurasi sebesar 64.43%, presisi sebesar 73.31%, dan recall sebesar 69.87% dengan waktu rata-rata eksekusi selama 25.72 detik. Untuk ukuran citra 200x200 dan *neighbors* 4x4, diperoleh hasil akurasi 63.62%, presisi 72.47%, dan recall 71.78% dengan waktu rata-rata eksekusi selama 11.63 detik. Untuk ukuran citra 128x128 dan *neighbors* 4x4, diperoleh hasil akurasi 63.31%, presisi 63.31%, dan recall 72.61% dengan waktu rata-rata eksekusi selama 3.05 detik. Untuk ukuran citra 256x256 dan *neighbors* 8x8, diperoleh hasil akurasi 65.59%, presisi 72.67%, dan recall 75.64% waktu rata-rata eksekusi selama 41.73 detik. Untuk ukuran citra 200x200 dan *neighbors* 8x8, diperoleh hasil akurasi 65.49%, presisi 71.98%, dan recall 75.79% dengan waktu rata-rata eksekusi 16.84 detik. Untuk ukuran citra 128x128 dan *neighbors* 8x8, diperoleh hasil akurasi 65.28%, presisi 65.28%, dan recall 78.55% waktu rata-rata eksekusi selama 3.75 detik. Berikut tabel hasil rangkuman eksperimen pengaruh ukuran citra dan *neighbors* terhadap akurasi segmentasi.

Tabel 4.1 Hasil Eksperimen ukuran citra dan neighbors

Image Size	Region Growing	Image/s	Avg. Accuracy	Avg. Precision	Avg. Recall
256x256	4x4	25.72	64.43%	73.31%	69.87%
200x200	4x4	11.63	63.62%	72.47%	71.78%
128x128	4x4	3.05	63.31%	63.31%	72.61%
256x256	8x8	41.73	65.59%	72.67%	75,64%
200x200	8x8	16.84	65.49%	71.98%	75.79%
128x128	8x8	3.75	65.28%	65.28%	78.55%

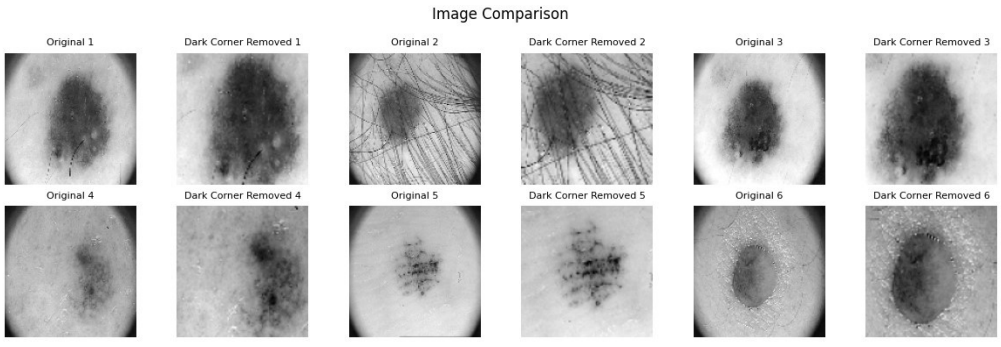
Berdasarkan hasil yang didapat, dapat diambil beberapa pertimbangan, Metode Region Growing dengan *neighbors* 8x8 pada Ukuran Citra 256x256 memiliki kinerja yang baik dengan akurasi, presisi, dan recall yang tinggi. Namun, waktu eksekusi yang lebih lama bisa menjadi kekurangan dalam aplikasi real-time atau ketika memproses citra dalam jumlah besar. Metode Region Growing dengan *neighbors* 4x4 pada Ukuran Citra 128x128 memiliki waktu eksekusi yang cepat dan memberikan hasil yang kompetitif dalam hal akurasi, presisi, dan recall. Ini bisa menjadi pilihan yang lebih baik dalam kasus di mana kecepatan eksekusi menjadi pertimbangan penting. Metode Region Growing dengan *neighbors* 8x8 pada Ukuran Citra 128x128 juga memberikan hasil yang baik dengan akurasi, presisi, dan recall yang sebanding dengan ukuran citra yang lebih besar (256x256), tetapi waktu eksekusinya lebih cepat. Atas hasil tersebut maka penelitian ini akan mengunakan ukuran citra 128x128 dan *neighbors* 8x8 dengan nilai akurasi 65.28%, presisi 65.28%, dan recall 78.55%. Hasil proses segmentasi dari setiap ukuran dan *neighbors* di lampirkan di lampiran I .

Tahap selanjutnya dari eksperimen segmentasi adalah menguji pengaruh preprocessing dengan nilai akurasi dari segmentasi region growing. Teknik preprocesing yang akan di coba diantaranya konversi citra ke grayscale , *dark corner removal*, *median filter* dan *Contrast Limited Adaptive Histogram Equalization (CLAHE).*

Pada penelitian kali ini citra dari *dataset* akan di konversi menjadi citra abu-abu, hal ini perlu dilakukan dikarenakan berdasarkan penelitian [5] segmentasi pada citra abu-abu menghasilkan hasil segmentasi lebih tinggi dari pada citra berwarna.

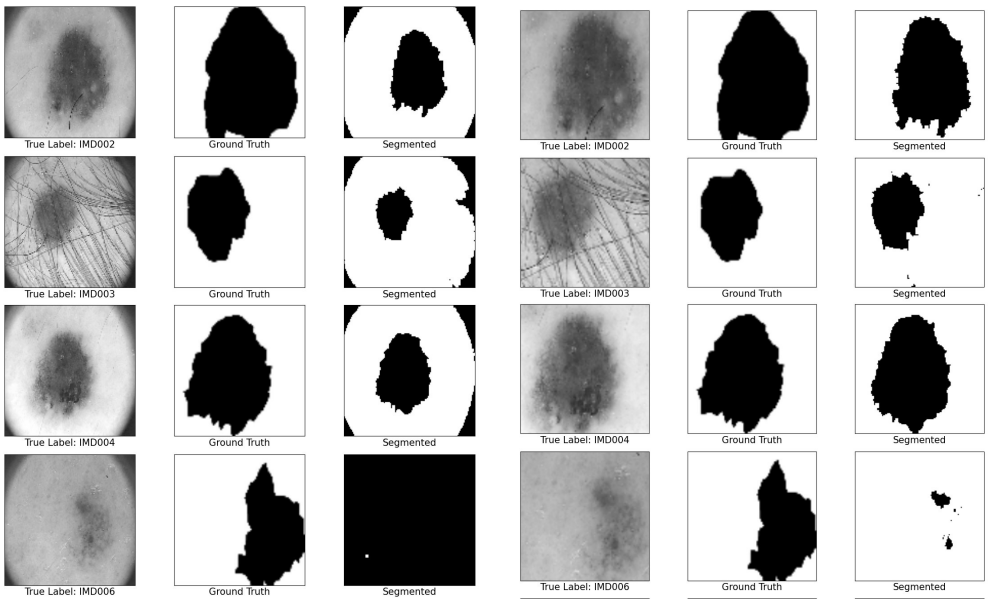
Teknik *Preprocessing* yang akan di uji pertama kali adalah penerapan modul *Corner borders removal* dikarenakan pada citra di dataset PH2 terdapat *object* yang cukup menganggu proses segmentasi yaitu *background* hitam disudut citra *dermoscopy*, dikarenakan hal tersebut penelitian ini melakukan *croping* gambar dengan memotong 20 *pixel* pada setiap sisi atas bawah dan kiri kanan. Setelah dipotong ukuran citra dikembalikan ke ukuran semula sebelum dipotong yaitu 128 x128. Hal ini telah dilakukan di penelitian [29] dan mampu mengurangi tingkat kesalahan deteksi.

hasil *croping image* mengunakan algorima *Corner borders removal* dapat dilihat pada gambar 4.2.



Gambar 4.2. Citra *dataset* sebelum dan sesudah di c*rop*

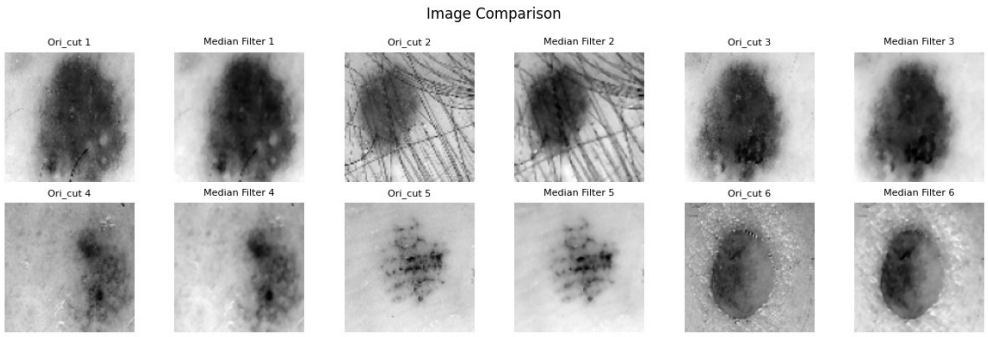
Setelah didapatkan data citra yang sudah di proses *dark corner removal* , langkah selanjutnya adalah melakukan segmentasi dengan data tersebut. Tidak lupa menjalankan segmentasi dengan data citra asli. Pengujian dilakukan dengan mengambil 40 citra dari *dataset* dengan ukuran citra 128x128 dan ukuran *neighbors* 8x8. Hasil Segmentasi dari citra asli mendapatkan nilai akurasi 61.9%, presisi 80% dan *recall* 67.6%. Sedangkan hasil segmentasi dari citra yang sudah di *corping* mendapatkan nilai akurasi 65.97%, presisi 63.94% dan recall 84.44%. Berdasarkan hal tersebut penelitian ini akan menggunakan teknik preprocesing *dark corner removal.* Berikut gambar 4.3 menunjukkan komparasi hasil segmentasi dari proses segmentasi antara citra asli dengan citra yang sudah di *croping* .



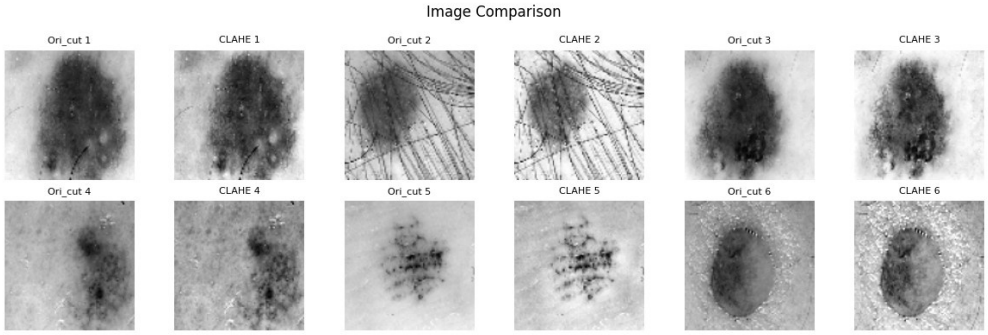
Gambar 4.3. Hasil Segmentasi Citra asli dan citra dari *dark corner removal*

Teknik *preprocessing* selanjutnya yang akan di uji adalah *median filter* dan *CLAHE*. implementasi *median filter* dilakukan dengan menggunakan modul

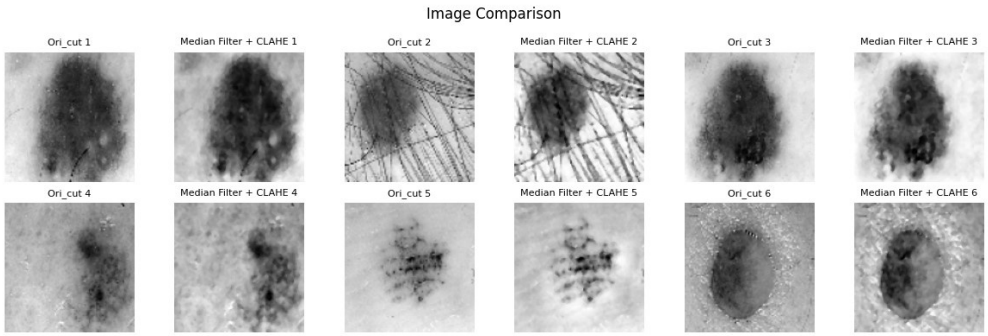
medianBlur dari *library openCV2* dan untuk *CLAHE* dilakukan dengan menggunakan modul *createCLAHE dari library openCV2.* Untuk implementasi *CLAHE* diperlukan nilai untuk parameter ukuran *tiles* untuk ukuran *histogram equalization* dan clip limit untuk mencegah kontras berlebihan. Berdasarkan penelitian [30] ukuran parameter *CLAHE* yang digunakan pada penelitian ini *tiles* 8 agar mendapatkan *contrast enhancement* yang lebih paling tajam dan ukuran *clip limit* 1% sebagai pembatas dari *contrast*. Penggunaan parameter tersebut membantu penelitian [30] mendapatkan akurasi klasifikasi yang cukup tinggi. Berikut gambar 4.4, 4.5 dan 4.6 terdapat hasil dari *preprocessing Median Filter* , *CLAHE* dan gabungan keduanya.



Gambar 4.4. Hasil Preprocessing *Median Filtered*

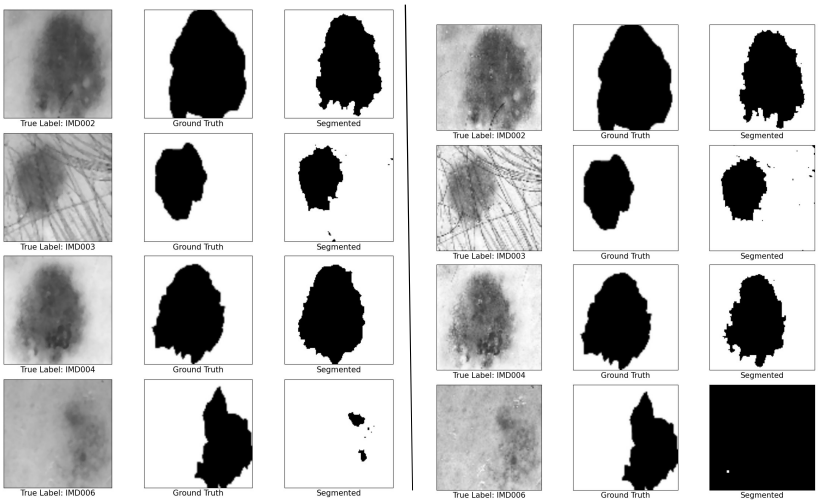


Gambar 4.5. Hasil Preprocessing *CLAHE*

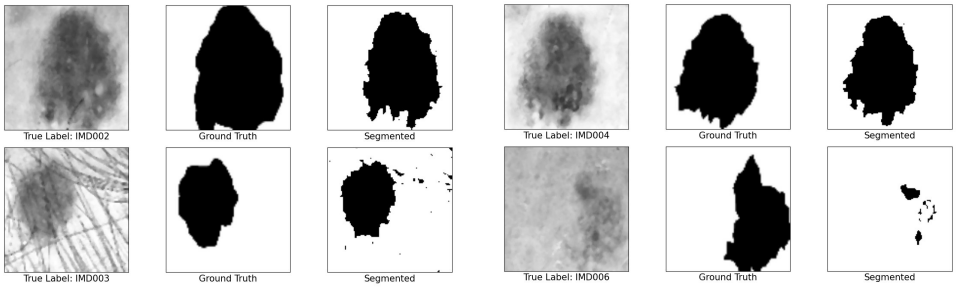


Gambar 4.6. Hasil Preprocessing *Median Filtered + CLAHE*

Citra hasil *preprocessing* diatas dilakukan uji segmentasi dengan *region growing* dan didapatkan untuk hasil segmentasi citra dari *preprocessing median filtered* nilai akurasi 65.41% , presisi 64.58% dan *recall* 84.05 % . Hasil uji segmentasi citra dari *preprocessing CLAHE* mendapatkan nilai akurasi 60.65% , presisi 62.24% dan *recall* 77.17%. Sedangkan untuk gabungan dari keduanya mendapatkan tingkat akurasi 70.19% , presisi 68.042 % dan *recall* 83.67%. Berikut gambar perbandingan dari hasil setiap segmentasi.



Gambar 4.7. Hasil Segmentasi Median Filtered (kiri) dan CLAHE (kanan)



Gambar 4.8. Hasil Segmentasi *Median Filtered* + *CLAHE*

Tabel 4.2 Hasil Eksperimen *Preprocessing* dan segmentasi *Region Growing*

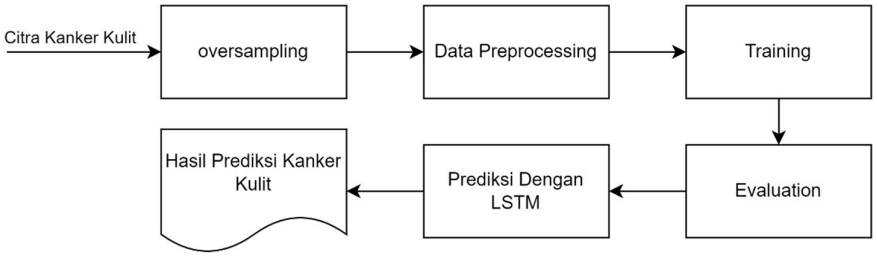
No.	Teknik Processing	Akurasi	Presisi	Recall
1	-	61,94%	80,64%	67,64%
2	Dark Corner Removal	65,98%	63,94%	84,45%
3	Dark Corner Removal + Median Filtered	65,42%	64,59%	84,05%
4	Dark Corner Removal + CLAHE	60,65%	62,24%	77,17%
5	Dark Corner Removal + Median Filtered + CLAHE	70,20%	68,04%	83,68%

Berdasarkan hasil segmentasi *Region Growing* pada tabel 4.2 didapatkan teknik processing terbaik adalah pengunaan *Dark Corner Removal* , *Median Filtered* dan *CLAHE*. Maka pengabungan 3 teknik *preprocessing* tersebut akan digunakan dalam penelitian ini dan hasil segmentasinya akan digunakan untuk *training* model LSTM.

4.2 Hasil Metode *LSTM*

Dalam eksperimen deteksi penyakit kanker kulit dengan metode LSTM, langkah-langkahnya adalah sebagai berikut: Pertama, dilakukan teknik *oversampling* untuk menyeimbangkan *dataset*. Kedua, dataset dibagi menjadi tiga bagian yaitu *training*, *testing*, dan *validation*. Selanjutnya, dilakukan *pra-pemrosesan* data seperti normalisasi, *resizing*, dan *preprocessing* data. Setelah itu, dilakukan eksperimen untuk menentukan ukuran citra yang optimal. Kemudian, dilakukan penyesuaian *hyperparameter* untuk mencari struktur model yang tepat. Selanjutnya, dilakukan uji *epoch* untuk mendapatkan jumlah *epoch* yang

baik sehingga tidak overfitting.. Terakhir, sebelum prediksi dilakukan *training* dengan data training dan *validation* dan dilakukan juga penilaian terhadap model baik dari akurasi , presisi dan recallnya . Dengan mengikuti langkah-langkah ini, eksperimen *LSTM* untuk deteksi penyakit kanker kulit dapat dilakukan secara sistematis dan diharapkan menghasilkan model prediksi yang akurat. Untuk informasi lebih detail mengenai proses eksperimen LSTM secara keseluruhan, dapat dilihat pada Gambar 4.1.



Gambar 4.9. Flowchart Eksperimen LSTM

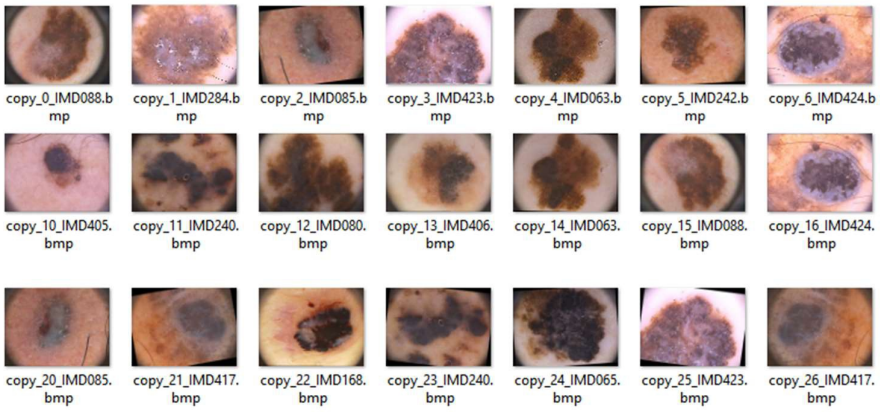
Sebelum memasuki tahap prediksi menggunakan LSTM, dilakukan tahap *oversampling* sebagai langkah pertama. Kemudian data dibagi menjadi partisi data training, data validasi, dan data pengujian dengan menggunakan *library Python* yang disebut *splitfolders*. Selain itu, karena *dataset* yang digunakan dalam penelitian ini tidak seimbang, seperti yang terlihat dalam tabel 4.3, maka jumlah data dalam setiap kelas perlu disesuaikan melalui *oversampling* atau *undersampling*.

Subfolder	Number of Images
Benign	80
Melanoma	40
Normal Skin	80

Tabel 4.1 : Hasil *SplitFolder* tanpa *OverSampling*

Dalam penelitian ini, metode *oversampling* digunakan. Hal ini didasarkan pada temuan penelitian [28] yang menyatakan bahwa teknik oversampling menghasilkan hasil yang lebih baik daripada undersampling. Oleh karena itu, dalam penelitian ini, teknik oversampling digunakan.

Dalam melakukan oversampling, penelitian ini melakukannya generate data di kelas minoritas dengan cara melakukan pengadaan data sampai mencapai jumlah data yang sama dengan kelas mayoritas. Demi mendapatkan hasil penelitian yang lebih baik data yang digandakan akan di rubah dahulu dengan proses *image augmentation*, yang dilakukan secara acak dapat berupa pembalikan citra secara horizontal atau vertikal atau merotasi citra (antara -15◦ hingga 15◦) atau pembesaran (10%-25%) sehingga data yang digandakan tidak sama dengan data yang sebelumnya. Berikut citra hasil *Oversampling* dan *Image Augmentation.*



Gambar 4.10. Citra hasil Oversampling dan Image Augmentation

Dari hasil proses *oversampling* data kelas minoritas (*melanoma*) memiliki jumlah data yang sama dengan kelas lainnya. Maka langkah selanjutnya adalah membagi data didataset untuk digunakan sebagai data *train*, *validation* dan *testing* untuk

proses *training* data dengan mengunakan *library splitfolders*. Penelitian ini akan menggunakan 16 data untuk *validation* dan 16 data testing dari setiap kelas dan sisanya untuk training. Inisiasi dari *splitfolders* yang digunakan pada penelitian ini adalah sebagai berikut:

$$splitfolders.fixed(input\_folder, output\_folder, -seed = 1337, fixed = (16,16), oversample = None, group\_prefix = None)$$

Setelahnya, dilakukan proses *splitfolders* untuk membagi *dataset* menjadi data training, data validasi, dan data testing. Hasil dari proses ini menunjukkan bahwa jumlah data untuk *dataset training* pada setiap kelas adalah sebanyak 48 data, dengan total data keseluruhan sebanyak 144. Selain itu, terdapat 16 data untuk *dataset* validasi dan 16 data untuk *dataset* testing. Tabel 4.4 menunjukkan hasil dari proses *splitfolders* di mana terlihat bahwa jumlah citra pada setiap kelas telah menjadi sama dan tidak ada lagi kelas yang memiliki citra lebih banyak dibandingkan kelas lainnya.

Benign		Melanoma		Normal Skin	
Data	Images	Data	Images	Data	Images
Train	48	Train	48	Train	48
Test	16	Test	16	Test	16
Validation	16	Validation	16	Validation	16

Tabel 4.4 Hasil splitfolders

Setelah direktori masing-masing kelas telah seimbang, proses *pre-processing* citra *dermoscopy* dapat dimulai. Pertama, citra *dermoscopy* dikonversi dari RGB ke *grayscale* menggunakan *library OpenCV* dengan menggunakan fungsi *imread*(). Selanjutnya, citra dikurangi ukurannya menggunakan fungsi *resize*() pada *OpenCV*.

Hal ini dilakukan untuk mempercepat proses *deep learning*, namun diperlukan eksperimen kecil untuk menentukan apakah ukuran gambar mempengaruhi akurasi model *deep learning*.

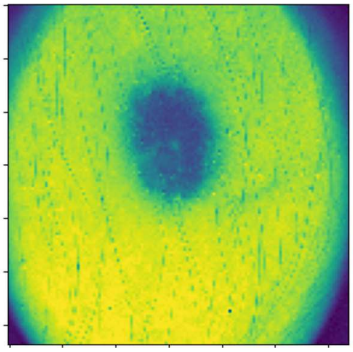
Hasil eksperimen dapat dilihat pada Tabel 4.5, merupakan hasil eksperimen untuk menentukan ukuran citra yang sesuai menggunakan model *deep learning* sederhana dengan 1 layer LSTM. Training dan validasi dilakukan selama 20 *epochs* dengan *batch size* 32. *Epochs* adalah jumlah iterasi yang dilakukan oleh model *deep learning*, dan penting untuk memperhatikan waktu eksekusi. Dalam eksperimen ini, jumlah epochs yang sama digunakan agar evaluasi dapat dibandingkan dengan validitas. Hasil menunjukkan bahwa akurasi validasi tidak berbeda signifikan, namun waktu yang dibutuhkan untuk training dan validasi sangat berkurang.

Tabel 4.5 : Hasil Eksperimen Ukuran Citra

No	Ukuran Gambar	Hasil				
		Training		*Validation*		Waktu Eksekusi
		Accuracy	*Loss*	*Accuracy*	*Loss*	
1	256 X 256	0.7521	0.6528	0.5208	0.9333	26.4 detik
2	200 X 200	0.7568	0.6042	0.4375	0.8840	19.1 detik
3	128 X 128	0.7014	0.7372	0.5417	0.8515	15.9 detik

Penelitian sebelumnya [31] juga menunjukkan bahwa ukuran citra tidak selalu berdampak pada akurasi model *deep learning*, dan waktu eksekusi cenderung meningkat dengan ukuran citra yang lebih besar. Berdasarkan eksperimen ini, dapat disimpulkan bahwa ukuran gambar 128x128 adalah yang paling tepat untuk penelitian ini.

Lalu gambar dikonversi menjadi *numpy array* dengan bantuan *library numpy* yaitu fungsi *array* () dan langkah terakhir adalah normalisasi warna pada gambar dari rentang 0 sampai 255 menjadi dari rentang 0 sampai 1, hasil pra-pemrosesan data dapat dilihat pada Gambar 4.6 sebagai salah satu contoh citra daun yang telah di proses sebelumnya untuk masuk ke dalam *deep learning* model.



Gambar 4.11: Hasil Data *Pre-processing*

Selanjutnya, dilakukan eksperimen untuk menentukan struktur model *deep learning*. Pada tahap ini, dilakukan perbandingan antara 2 model, yaitu 2 *layer LSTM* dan 1 layer LSTM. Selain itu, juga dilakukan *tuning hyperparameter* menggunakan *library Keras Tuner*. *Keras Tuner* digunakan dengan algoritme *random search* untuk mencari parameter terbaik untuk model deep *learning* LSTM. Parameter yang *dituning* adalah jumlah neuron pada layer LSTM dan nilai *learning* rate yang sesuai. Proses *tuning hyperparameter* ini dibagi menjadi 2 bagian, yaitu dari nilai node 32 hingga 128 dan dari 128 hingga 256. Untuk *learning rate*, pilihan yang diberikan kepada tuner adalah 0.01, 0.001, dan 0.0001.

Proses *tuning* dilakukan dengan tujuan mengurangi variabel *validation loss*. Selain itu, digunakan juga teknik *early stopping* untuk menghentikan proses *tuning* setelah validation loss tidak berkurang selama 10 *epoch*. Setiap iterasi *tuning* dilakukan selama 100 *epoch*, dan keseluruhan proses tuning ini diulang sebanyak 3 kali. Total iterasi tuning yang dilakukan adalah 5 kali.

Hasil dari eksperimen dengan menggunakan 1 *layer LSTM* dapat dilihat pada tabel 4.6. Di Eksperimen *Tuner* 1 Layer LSTM (32-128) mendapatkan hasil model terbaik dengan menghasilkan skor *validation loss* terendah sebesar 0.8417290449142456 adalah dengan mengunakan *node LSTM* sebesar 96, dengan *learning rate* 0.0001. Pada 1 Layer LSTM (128-256) hasilnya adalah skor *validation loss* terendah sebesar 0.8408656318982443 adalah dengan mengunakan *node LSTM* sebesar 144, dengan *learning rate* 0.0001 .

Tuner	1 Layer LSTM (32-128)	1 Layer LSTM (128-256)
input_unit	96	144
learning_rate	0.0001	0.0001
Score	0.8417290449142456	0.8408656318982443

Tabel 4.6 : Hasil Eksperimen 1 Layer LSTM

Selanjutnya, dilakukan eksperimen untuk menentukan nilai *validation loss* terbaik pada model dengan dua layer *LSTM*. Proses eksperimen ini mirip dengan eksperimen sebelumnya untuk model satu layer *LSTM*, tetapi perbedaannya terletak pada jumlah layer *LSTM* yang digunakan. proses eksperimen dibagi menjadi dua

bagian: pertama, mencari jumlah node *LSTM* di antara 32 hingga 128; dan kedua, mencari jumlah *node LSTM* di antara 128 hingga 256.

Hasil dari eksperimen dua layer *LSTM* dapat dilihat pada Tabel 4.7. Pada Tuner 2 Layer LSTM (32-128) di mana model terbaik ditemukan dengan jumlah *node LSTM* pertama sebanyak 64, jumlah node LSTM kedua sebanyak 112, dan nilai *learning rate* sebesar 0.001. Nilai *validation loss* terendah yang ditemukan adalah 0.8340716361999512. Selanjutnya pada 2 Layer LSTM (128-256) di mana model terbaik ditemukan dengan jumlah node *LSTM* pertama sebanyak 128, jumlah *node LSTM* kedua sebanyak 240, dan nilai *learning rate* sebesar 0.0001. Nilai *validation loss* terendah yang ditemukan adalah 0.8432829777399699.

Tuner	2 Layer LSTM (32-128)	2 Layer LSTM (128-256)
input_unit	64	128
layer_2_neurons	112	240
learning_rate	0.001	0.0001
Score	0.8340716361999512	0.8432829777399699

Tabel 4.7 : Hasil Eksperimen 2 *Layer LSTM*

Rangkuman hasil eksperimen *hyperparameter tuning* dapat dilihat pada Tabel 4.2.

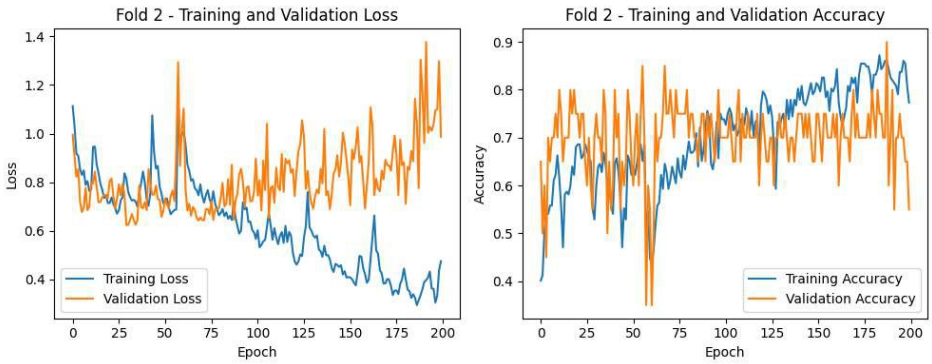
Tabel 4.8 Eksperimen Hyperparameter Tuning

No	Jumlah *Layer*	Rentang *nodes*	Jumlah Node		Hasil	
			Node layer 1	*Node layer 2*	*Learning rate*	*Validation loss*
1	1	32-128	96	-	0.0001	0. 841729
2		128-256	144	-	0.0001	0. 840865
3	2	32-128	64	112	0.001	0. 834071
4		128-256	128	240	0.0001	0. 843282

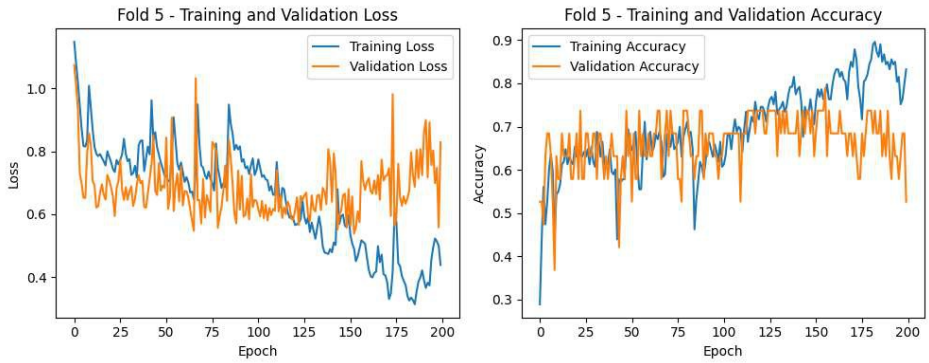
Berdasarkan hasil *hyperparameter tuning* pada **Error! Reference source not found.** didapat struktur model terbaik yaitu banyaknya *layer LSTM* beserta jumlah *nodes* yang sesuai yaitu 2 *layer LSTM* dengan 64 nodes layer 1 dan 112 nodes layar 2. Sehingga model *training* yang akan digunakan pada penelitian ini adalah:

```
model = keras.Sequential()
model.add(keras.layers.LSTM(64, return_sequences=True))
model.add(keras.layers.LSTM(112))
model.add(keras.layers.Dense (3, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',
optimizer=keras.optimizers.Adam(learning_rate=0.001),metrics=['accuracy'])
```

Langkah selanjutnya adalah penentuan jumlah epoch yang akan digunakan untuk menjalankan model. maka dijalankan pengujian model sebanyak 10 kali dengan mengunakan epoch 200 dan di dapatkan *plot* akurasi dan loss berdasarkan jumlah *epoch.* Berikut di gambar 4.12 dan 4.13 terdapat *plot* akurasi dan *loss* dari pengujian ke-2 dan ke-5. plot lainnya dan tabel hasil uji dapat dilihat di lampiran II.



Gambar 4.12*:* Plot Akurasi dan *loss* dari uji *epoch* 200 pengujian ke-2



Gambar 4.13*:* Plot Akurasi dan *loss* dari uji *epoch* 200 pengujian ke-5

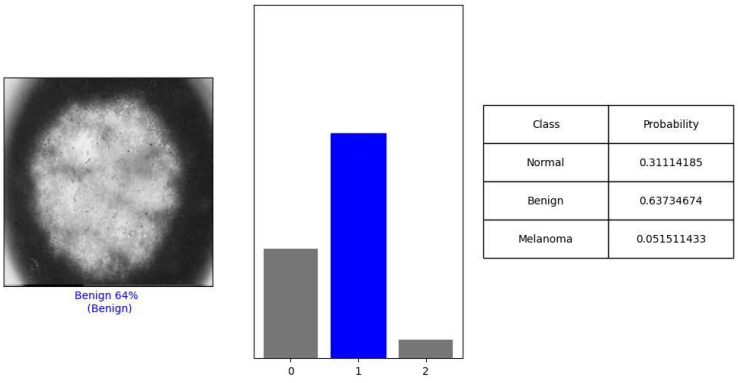
Dari Gambar *plot* hasil uji *epoch* 200 dapat dilihat rata-rata setelah epoch ke 25 nilai validasi loss meningkat. Peningkatan *validation loss* disebabkan oleh *overfitting*. Berdasarkan hasil uji epoch 200 maka jumlah epoch yang akan digunakan untuk penelitian ini adalah 25.

Setelah menentukan jumlah *epoch* dan struktur model *deep learning*, maka selanjutnya dilakukan tahap *training* menggunakan data *training* dan data validasi untuk menghasilkan sebuah model yang nantinya akan digunakan untuk memprediksi data testing. Model hasil tahap *training* perlu di evaluasi untuk mendapatkan tingkat akurasi , presisi dan *recall* nya dan didapatkan hasil dari model *LSTM* adalah 54.1% untuk akurasi , presisi sebesar 57.8% dan recall 54.1%.

Parameter	Value
Accuracy	0.5416666666666666
Precision	0.578088578088578
Recall	0.5416666666666666

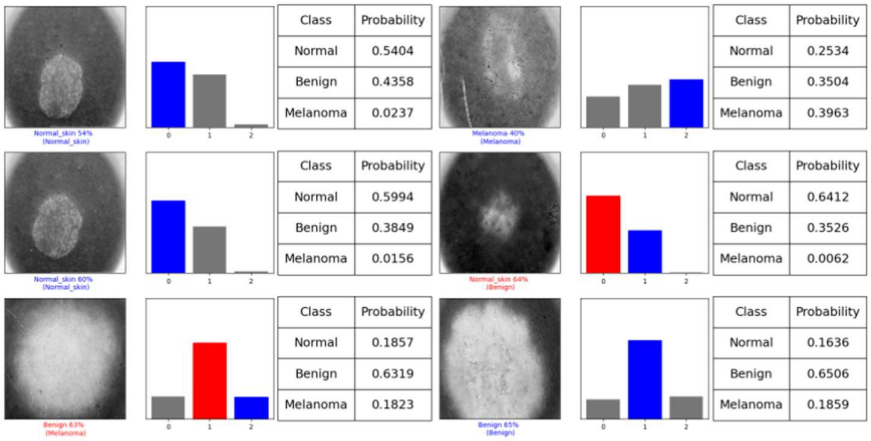
Tabel 4.9*:* Nilai akurasi , presisi dan *recall* dari model *LSTM*

Proses prediksi dilakukan untuk satu data dan seluruh data yang terdapat dalam *dataset testing*. Hasil prediksi ini dapat dilihat pada Gambar 4.14, yang menunjukkan contoh prediksi untuk citra *dermoscopy* kulit yang terkena penyakit kanker kulit kategori *Benign*. Jika model berhasil memprediksi bahwa citra *dermoscopy* kulit tersebut terkena penyakit, maka di bawah hasil prediksi akan ditampilkan saran terhadap pasien. Proses prediksi untuk seluruh data dalam dataset testing dilakukan dalam waktu 4.4 detik, dan hasil prediksinya dapat ditemukan pada lampiran III.



Gambar 4.14*:* Hasil Prediksi 1 Data

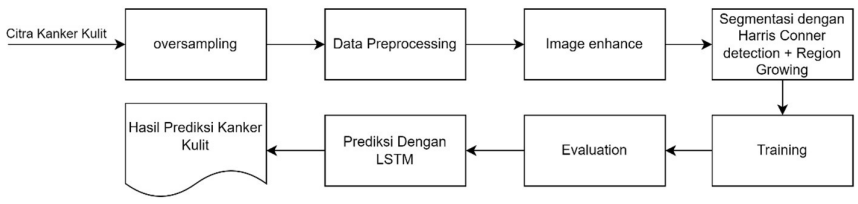
Selanjutnya dilakukan pengambilan sampel data dari dataset *testing* untuk dianalisis hasil prediksi dari algoritme *LSTM* yang dapat dilihat pada Gambar 4., dimana pada gambar dapat dilihat algoritme *LSTM* dapat memprediksi citra dari kelas *Normal Skin* dengan baik, namun untuk kelas *Benign* dan *melanoma* model ini masih kurang akurat dalam memprediksinya.



Gambar 4.15*:* Hasil Prediksi Sampel Dataset *Testing*

4.3 Hasil Metode *LSTM* dan *Region Growing*

Dalam eksperimen deteksi kanker kulit dengan menggunakan *LSTM* dan *Region Growing* parameter ukuran citra, model dan epoch yang digunakan sama seperti yang digunakan di implementasi metode *LSTM.* Hal yang membedakan adalah penggunaan data train , validation dan testing yang sudah di segmentasi dengan *Region Growing*. Data yang tersegmentasi akan di *training* dengan model *LSTM 2 layers* (64 dan 122) dan jumlah *epoch* 25. Model hasil training kemudian di evaluasi untuk mendapatkan nilai akurasi, presisi dan *recall*-nya. Diharapkan dengan menggunakan *dataset* yang sudah tersegmentasi akan meningkatkan akurasi prediksi yang lebih baik daripada menggunakan metode *LSTM* saja. Berikut informasi lebih detail mengenai proses eksperimen *LSTM* dan *Region Growing* secara keseluruhan, dapat dilihat pada Gambar 4.16.



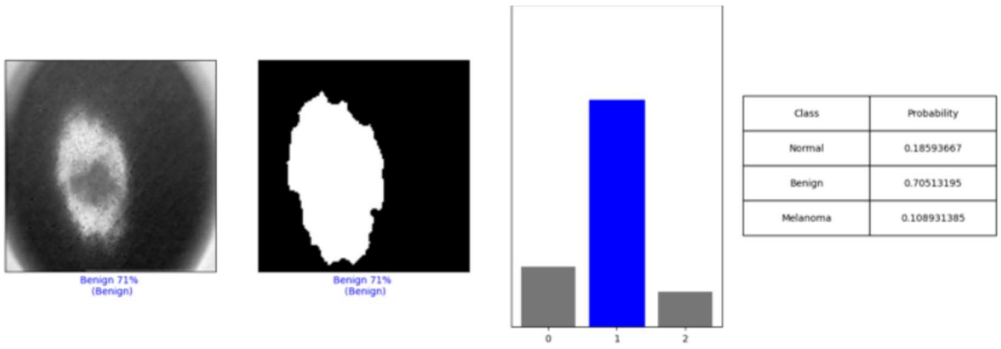
Gambar 4.16*:* Flowchart Eksperimen *LSTM* dan *Region Growing*

Pada proses oversampling mengunakan impementasi yang sama penerapannya dengan subbab 4.1 Hasil Metode *LSTM* . Setelah dioversampling data dilakukan proses preprocessing, image enhance dan segmentasi yang sama penerapannya dengan subbab 4.1 hasil Segmentasi Metode Region Growing. Setelah *dataset* tersegmentasi data tersebut ditraining dengan mengunakan model model *LSTM 2 layers* (64 dan 122) dengan jumlah *epoch* 25 . Hasil evaluasi dari model *LSTM* dan *Region Growing* mendapatkan nilai akurasi sebesar 75% , presisi sebesar 80.05 % dan *recall* sebesar 75%. Berikut pada gambar 4.23 terdapat hasil evaluasi metode *LSTM* dan *Region Growing.*

Parameter	Value
Accuracy	0.75
Precision	0.8051282051
Recall	0.75

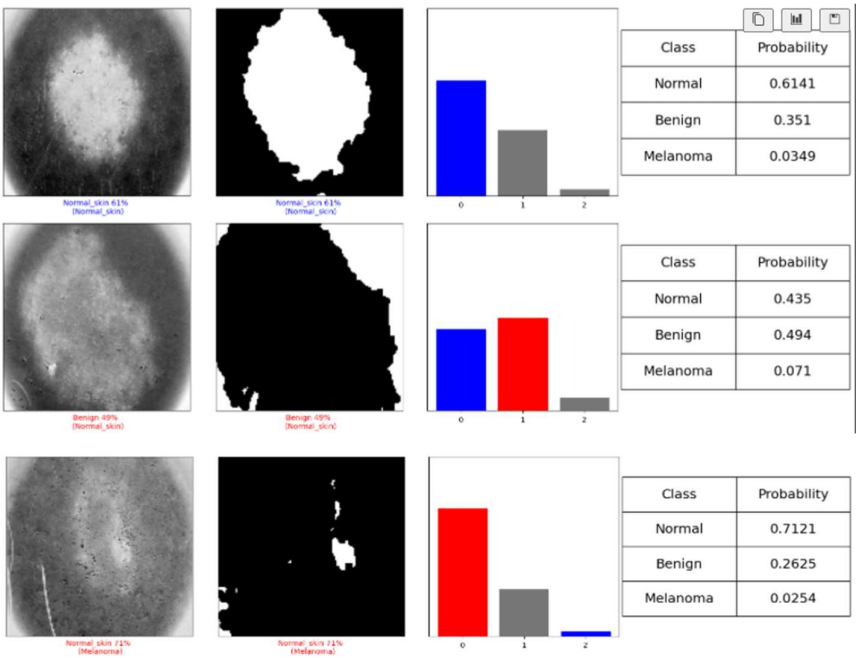
Tabel 4.10: Hasil evaluasi Model LSTM & Region Growing

Proses prediksi dilakukan untuk satu data dan seluruh data yang terdapat dalam *dataset testing*. Hasil prediksi ini dapat dilihat pada Gambar 4.24, yang menunjukkan contoh prediksi untuk citra *dermoscopy* kulit yang terkena penyakit kanker kulit kategori *Benign*. Jika model berhasil memprediksi bahwa citra *dermoscopy* kulit tersebut terkena penyakit, maka di bawah hasil prediksi akan ditampilkan saran terhadap pasien. Proses prediksi untuk seluruh data dalam dataset testing dilakukan dalam waktu 3.5 detik, dan hasil prediksi lainnya dapat ditemukan pada lampiran IV.



Gambar 4.17*:* Hasil Prediksi 1 Data LSTM & Region Growing

Selanjutnya dilakukan pengambilan sampel data dari dataset *testing* untuk dianalisis hasil prediksi dari algoritme *LSTM* yang dapat dilihat pada Gambar 4.25 , dimana pada gambar dapat dilihat algoritme *LSTM* dapat memprediksi citra dari kelas *Normal Skin* dengan baik, namun untuk kelas *Benign* dan *melanoma* model ini masih kurang akurat dalam memprediksinya.



Gambar 4.18*:* Hasil Prediksi Sampel Dataset *Testing*

4.4 Analisis Hasil dan Perbandingan Algoritme

Tabel perbandingan metode segmentasi pada penelitian ini dan penelitian terkait sebelumnya dapat dilihat di tabel 4.3. untuk hasil dari segmentasi *dark corner removal* dan *Region Growing* yang di bantu dengan *image enchament Median filter & CLAHE* serta *Harris Corner Detection* untuk penentuan seed belum bisa mendapatkan hasil akurasi segmentasi seperti penelitian [5] yang menerapkan *Harris Corner Detection + Region Growing + Post Processing filling and dilation* dikarenakan pengunaan ukuran citra yang berbeda. Faktor pertama adalah pengunaan citra pada penelitian ini lebih kecil sehingga tidak bisa mengunakan teknik *Post Processing filling and dilation* dengan seperti pada penelitian [5] . penelitian ini tidak bisa mengunakan citra dengan ukuran 523 × 382 pixels dikarenakan keterbatasan komputasi dari alat penelitian yang digunakan di penelitan ini. Faktor kedua *initial seed* dari hasil deteksi sudut dari *Harris Corner Detection* terkadang tidak tepat menitikan pada jejas atau luka di citra dermoscopy kulit hal ini bisa disebabkan object yang mengganggu di citra seperti rambut atau *dark corner* yang masih ada walau sudah di hapus oleh modul *dark corner removal*. Dan pada citra dengan jejas kulit samar-samar masih sulit untuk akurat meski sudah ditingkatkan kontrasnya dengan *CLAHE* Diperlukan teknik preprocessing lain yang bisa menghilangkan *object* yang tidak dibutuhkan untuk proses segmentasi. Serta *image enhacement* lainnya untuk meningkatkan detail citra yang memiliki jejas kulit yang tipis atau samar-samar.

Tabel 4.11 : Perbandingan Teknik Segmentasi & Processing

No.	Ukuran Citra	Teknik Segmentasi & Processing	Akurasi Segmentasi
1	128 x128	Dark Corner Removal + Median Filtered + CLAHE + Harris Corner Detection+Region Growing (proposed)	70,20%
2	523 × 382	Harris Corner Detection + Region Growing + Post Processing Flling and Dilation [5]	95,00%

Dari hasil eksperimen *training* dengan mengunakan metode *LSTM* dengan *Region Growing + LSTM* penerapan segmentasi sebelum *training* dapat meningkatkan tingkat akurasi model yang dibuat. Peningkatan akurasi sebesar 20.9% didapatkan setelah penerapan segmentasi di dataset. Peningkatan akurasi juga diikuti dengan peningkatan pada presisi dan recall. Hal ini terjadi dikarenakan segmentasi membantu proses *training* dengan memberikan citra yang lebih fokus ke jejas kulit dan menghilangkan object yang tidak perlu. Tabel 4.4 menunjukkan perbandingan hasil evaluasi metode *LSTM* dengan *Region Growing + LSTM* .

Tabel 4.12: Perbandingan Evaluasi LSTM dengan Region Growing + LSTM

	Accuracy	Precision	Recall	Train Durations	Predict Durations
LSTM	54%	57,8%	54%	39,3s	3,2s
Region Growing + LSTM	75%	80,5%	75%	16m 17,4s + 43,2 s	3m 46,4s + 3,5s

Tingkat akurasi klasifikasi 75% yang di hasilkan dari *Region Growing + LSTM* dari penelitian ini masih kecil jika dibandingkan dengan hasil penelitian sebelumnya yaitu penelitian [6], [15] dan [30] . Dapat dilihat di Tabel 4.5 akurasi dari penelitian lain mencapai 90% lebih.

Tabel 4.13: Perbandingan metode yang diajukan dengan penelitian sebelumnya

Image Processing	Dataset	Algorithm	Accuracy	Precision	Recall
Dark Removal , Median Filter, Clahe, Harris Corner Detection , Region Growing (Proposed)	PH2	LSTM	75,00%	80,50%	75,00%
Median filter, K-mean [6]	ISIC	RNN	93,00%	92,00%	88,00%
Morphological Black Hat Transformation, CLAHE [30]	HAM10000	CNN	87,99%	88,47%	97,98%
CLAHE [15]	ISIC	VGG-16	92.6%	-	-

Beberapa faktor mengapa penelitian lain bisa mendapatkan akurasi tinggi dan penelitian ini belum maksimal diantaranya : Faktor Jumlah Citra Dalam Dataset , Citra PH2 yang di gunakan pada penelitian ini hanya memiliki 200 citra sedangkan dataset yang digunakan penelitian [6], [15] dan [30] adalah *dataset HAM10000* yang memiliki 10000 data, dan *ISIC 2019* dataset yang memiliki jumlah data 20771. Dari perbedaan jumlah dataset yang begitu besar akan lebih banyak data yang bisa dipelajari dengan *LSTM* dan tidak cepat *overfitting* meski dengan *bacth* proses yang besar. *Dataset HAM10000* pun hanya mememiliki 2 kelas yaitu *benign* dan *malignan* sehingga klasifikasi dilakukan lebih mudah. Dataset *ISIC 2019* memiliki jumlah kelas yang banyak yaitu 8 akan tetapi tidak ada kelas yang menunjukkan kulit sehat atau minimal nevus pigmentosus (tahi lalat) seperti yang ada di ph2 dataset. Faktor kedua ialah akurasi dari implementasi segmentasi di penelitian ini masih belum maksimal , meski sudah dilakukan *preprocessing image enhancement* hanya mendapatkan 70% akurasi segmentasi optimasi teknik *preprocessing* dan *post-processing* diperlukan untuk peningkatan akurasi segmentasi, sehingga dataset yang disegmentasi semakin bernilai saat digunakan pada tahap *training*.