BAB 3

METODE DAN PERANCANGAN PROGRAM

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam mengklasifikasi tipe perdarahan otak yang terjadi menggunakan *Siamese Convolutional Neural Network* berdasarkan pada citra CT *scan* otak adalah sebagai berikut:

1. Studi Literatur

Tahap pertama, yaitu studi literatur dengan melakukan pencarian, pembelajaran dan pemahaman terhadap konsep klasifikasi citra, bahasa pemrograman yang akan digunakan, teori-teori yang berkaitan dengan *Siamese convolutional neural network* maupun penyakit perdarahan otak melalui *e-journal*, buku, *e-book* dan sumber referensi lainnya. Tahap ini bertujuan untuk meningkatkan pemahaman secara teori mengenai *Siamese Convolutional Neural Network* dan klasifikasi citra.

2. Pencarian Dataset

Dataset yang akan digunakan dalam penelitian ini, didapat dari suatu organisasi nirlaba dengan lebih dari 52.000 anggota dari 153 negara di seluruh dunia yang bernama *The Radiological Society of North America* (RSNA) bekerja sama dengan anggota *American Society of Neuroradiology* dan MD.ai. Dataset yang diambil berisikan citra *CT scan* otak dalam bentuk *Digital Imaging and Communication in Medicine* (DICOM) yang terdiri dari 5 tipe perdarahan otak dan data *csv* yang berisikan ID *patient* dan label yang menunjukkan ada atau tidaknya perdarahan otak.

3. Analisis dan Perancangan Program

Pada tahap analisis dan perancangan program, akan dilakukan dengan merancang alur kerja dalam mengklasifikasikan tipe perdarahan otak menggunakan *Siamese Convolutional Neurak Network* dan arsitektur *EfficientNet-B3*. Adapun perancangan antarmuka untuk memenuhi kebutuhan dalam penelitian ini.

4. Implementasi

Implementasi pada penelitian ini dilakukan dengan membuat *source code* untuk mengimplementasikan model *Siamese Convolutional Neural Network* untuk klasifikasi tipe perdarahan otak menggunakan Bahasa pemrograman Python dan memanfaatkan *library* KERAS. Implementasi akan terdiri dari beberapa proses, antara lain:

a. Preprocessing Data

Pada tahap *preprocessing*, dilakukan 2 *preprocessing* data, yaitu terhadap data csv dan data *image*. Pada data csv, akan dilakukan analisis fitur-fitur yang dimiliki dan membagi data menjadi 6 *dataframe* sesuai dengan kelasnya (healthy, epidural, subdural, subarachnoid, intraparenchymal dan intraventricular). *Preprocessing* juga dilakukan pada data *image* dengan hanya mengambil 3000 data untuk setiap kelas dan mengubah data *image* yang awalnya merupakan *DICOM file* menjadi PNG dengan menggunakan metode *CT Window* yang mentransformasikan nilai *pixel* menjadi *Hounsfield Units* (HU) untuk mendapatkan 3 area spesifik pada *image*, yaitu *brain window*, *bone window* dan *subdural window*.

b. Training

Perancangan model arsitektur *EfficientNet-B3* dibangun pada tahap ini memanfaatkan *library* KERAS. Model arsitektur ini sebagai *pre-trained model* yang digunakan untuk mengekstraksi fitur-fitur hasil *preprocessing* data *image* yang berukuran image 300x300.

Hasil *feature extraction* ini, akan dibagi menjadi data *train* dan data *test* yang nantinya akan disiapkan dalam bentuk *triplet pair* sebagai masukkan dari model *Siamese Convolutional Neural Network*. Keluaran dari model *Siamese Convolutional Neural Network* ini akan menjadi masukkan untuk melakukan klasifikasi tipe perdarahan otak dengan menggunakan KNN.

c. Testing

Tahap ini bertujuan untuk menguji model yang telah diimplementasikan telah berjalan dengan baik atau belum. Proses *testing* akan menggunakan data *test* yang kemudian akan dilakukan evaluasi performa dengan menghitung *accuracy*, *precision*, *recall*, dan *F1-score* menggunakan *confusion matrix* dari klasifikasi tipe perdarahan otak yang telah dilakukan.

d. Uji Coba

Uji coba akan dilakukan untuk menganalisis dan mengevaluasi penelitian yang telah dilakukan, sehingga hasil penelitian ini dapat dilihat kelayakannya.

e. Kesimpulan

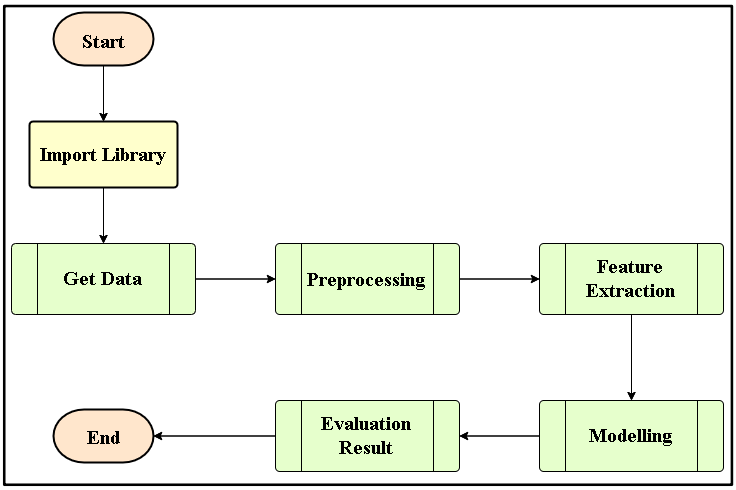
Penarikan kesimpulan dari implementasi dan data yang telah diuji coba berdasarkan rumusan masalah dan tujuan penelitian. Penarikan kesimpulan akan berupa persentase tingkat akurasi untuk melihat kinerja sistem yang telah diteliti ini.

5. Penulisan laporan

Penulisan laporan didasari pada penelitian yang telah dilakukan sebagai dokumentasi dari penelitian yang telah dilakukan. Tahap ini berjalan bersamaan dengan tahapan lainnya.

3.2 Perancangan Program

Perancangan program untuk klasifikasi tipe perdarahan otak dengan menggunakan metode *Siamese Convolutional Neural Network* secara keseluruhan digambarkan dengan diagram alur atau *flowchart* yang ditunjukkan pada Gambar 3.1.

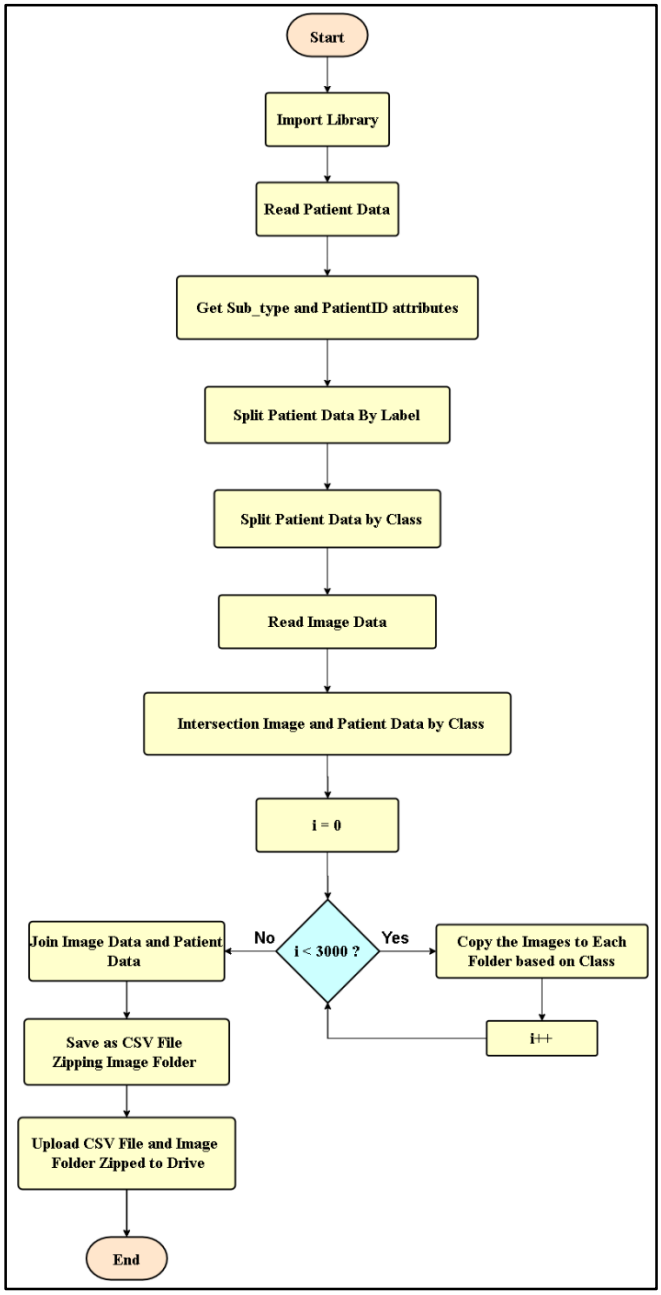


Gambar 3.1 *Flowchart* Keseluruhan

Program terdiri dari 5 proses utama, yaitu *get data*, *preprocessing*, *feature extraction*, *modelling* dan *evaluation result*. Sebelum masuk ke proses utama, dilakukan *import library* yang diperlukan terlebih dahulu untuk mendukung proses penelitian agar lebih efektif.

3.2.1 Get Data

Gambar 3.2 merupakan *flowchart* atau alur kerja dari proses mendapatkan data (*Get Data*). Data yang akan digunakan adalah dataset *brain hemorrhage* dari RSNA yang berisi citra *CT Scan* otak dalam file dan data pasien terkait.



Gambar 3.2 Flowchart Get Data

Pada proses ini, dilakukan pengambilan data baik data citra maupun data pasien yang akan digunakan dalam penelitian

A. Data Citra

Proses pengambilan data citra *CT Scan* otak dilakukan dengan mengelompokkan dan memisahkan penyimpanan citra per folder berdasarkan kelasnya. Pengambilan akan dilakukan sebanyak 3000 data citra *CT Scan* otak untuk masing-masing folder kelas yang menunjukkan tipe perdarahan, yakni Epidural, Subdural, Subarachnoid, Intraparenchymal dan Intraventricular serta kelas Healthy yang berisikan data citra otak dimana tidak terjadi perdarahan. Data citra selanjutnya akan di-*zip* dan di-*upload* ke *google drive* untuk digunakan pada proses penelitian selanjutnya.

B. Data Pasien

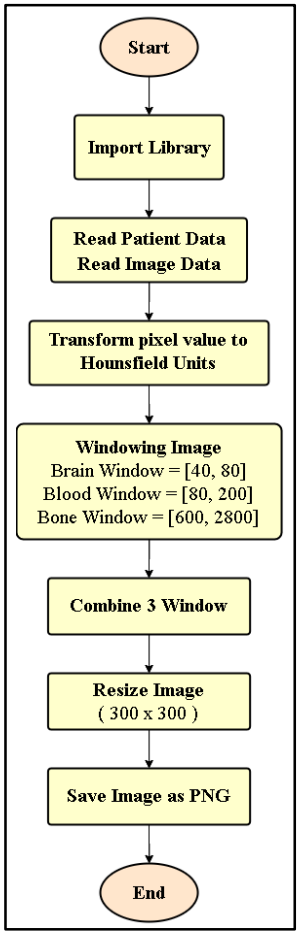
Pada data pasien dilakukan pemisahan, terhadap atribut *ID* menjadi 2 atribut baru, yakni atribut *Sub-type* (tipe perdarahan) dan atribut *PatientID*. Pemisahan juga dilakukan pada data pasien berdasarkan label yang kemudian dikelompokkan kembali berdasarkan masing-masing kelas. Pengambilan data pasien menyesuaikan 3000 citra yang telah diambil sebelumnya sehingga akan terambil pula 3000 data pasien untuk 6 kelas, sehingga data pasien berjumlah 18.000 data. Data pasien tersebut akan di-*upload* ke *google drive* untuk digunakan pada proses selanjutnya.

3.2.2 Preprocessing

Proses *preprocessing* data dilakukan untuk keperluan klasifikasi dengan harapan dapat mencapai performa yang lebih baik serta tepat sasaran. Tahapan *preprocessing*

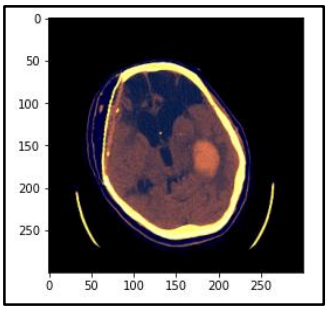
dilakukan dengan menggunakan metode *CT Window* untuk mengkonversikan citra *CT Scan* yang awalnya memiliki ekstensi DICOM menjadi PNG.

Tahap awal yang dilakukan, yaitu mentransformasikan nilai *pixel* dari citra menjadi *Hounsfield Units* (HU). Transformasi nilai *pixel* tersebut memanfaatkan nilai *Rescale Intercept* dan *Rescale Slope* yang merupakan fitur dari citra *CT Scan*. Tahap berikutnya, dilakukan *windowing* terhadap citra *CT Scan* untuk mendapatkan *specific zone* dari citra tersebut. Berhubung akan menggunakan arsitektur *EfficientNet-B3*, maka *windowing* akan dilakukan pada 3 area. Tiga area tersebut antara lain: area otak (*Brain Window*), area perdarahan (*Blood Window*) dan area tulang (*Bone Window*).



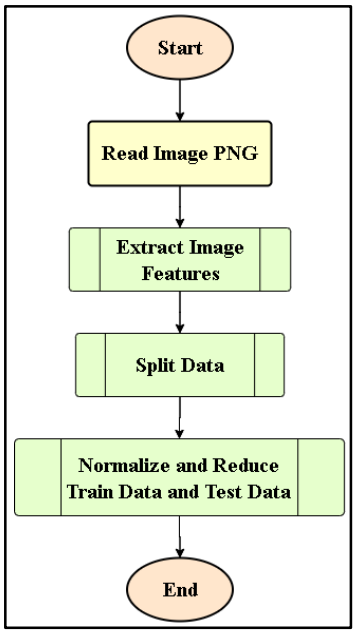
Gambar 3.3 Flowchart Preprocessing

Ketiga hasil *windowing* tersebut akan digabungkan membentuk suatu array dengan ordo 3x3 ([*brain window*, *blood window*, *bone window*]) yang kemudian dilakukan normalisasi sehingga nilai hanya berada pada rentang 0 sampai 1. Pada tahap akhir, dilakukan perubahan ukuran citra menjadi 300x300 sebelum disimpan dalam ekstensi PNG untuk menyesuaikan *input shape* dari arsitektur *EfficientNet-B3*. Hasil akhir preprocessing citra *CT Scan* akan terlihat seperti pada Gambar 3.7.



Gambar 3.4 Hasil Preprocessing (Epidural Image)

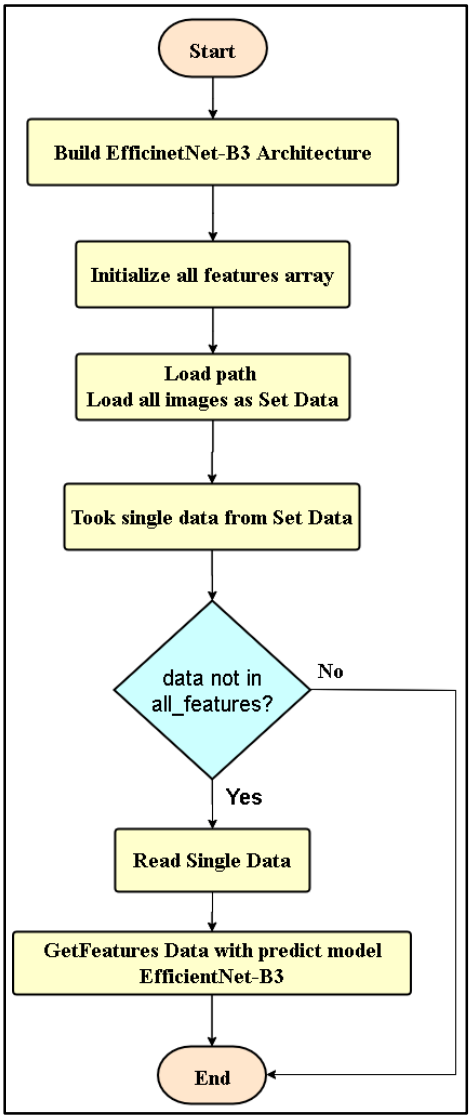
3.2.3 Feature Extraction



Gambar 3.5 Flowchart Feature Extraction

Proses *feature extraction* dimulai dari membaca data citra hasil *preprocessing* yang telah disimpan sebelumnya, kemudian diikuti dengan 3 sub-proses, antara lain:

A. Extract Image Features

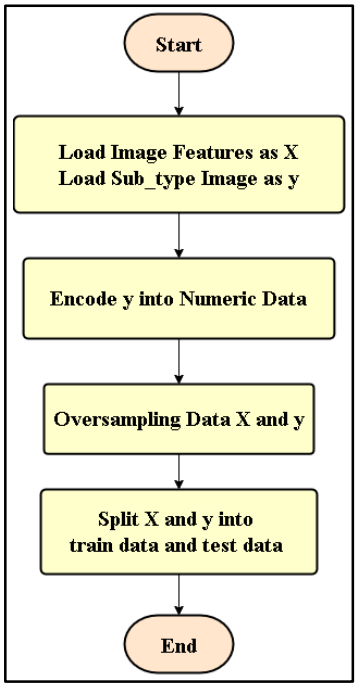


Gambar 3.6 Flowchart Extract Image Features

Gambar 3.6 menunjukkan *flowchart* atau alur kerja ekstraksi fitur. Ekstraksi fitur citra dilakukan untuk mendapatkan fitur-fitur dari setiap citra pada setiap kelas. Pada penelitian ini, fitur suatu citra berisikan hasil prediksi model dengan menggunakan

model *EfficientNet-B3* dengan masukkan data citra yang berukuran 300x300. Pembangunan model arsitektur *EfficientNet-B3* menggunakan *class efficientNet* dari *library* KERAS dengan mengatur parameter *weights* = '*imagenet*' dan ditambahkan GlobalAveragePooling2D untuk meminimalisir resiko terjadinya *overfitting*.

B. Split Data



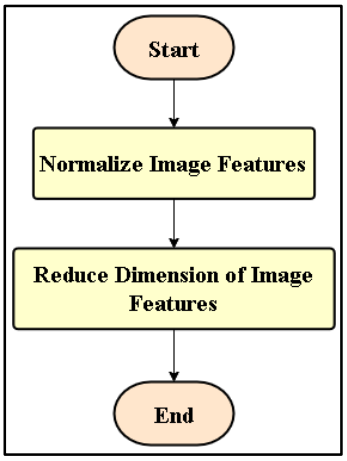
Gambar 3.7 Flowchart Split Data

Hasil ekstraksi fitur citra dimasukkan ke dalam 2 variabel X dan y, dimana X berisikan fitur-fitur citra setiap pasien dan y berisikan *sub_type* dari pasien tersebut. *Sub_type* pada awalnya merupakan data kategorik diubah menjadi ke dalam bentuk *numeric* dengan memanfaatkan *class* dari *library* Sklearn, yaitu *LabelEncoder*. Berikutnya, akan dilakukan oversampling data untuk memperbanyak jumlah data. Langkah terakhir dilakukan pembagian data ke dalam *train data* dan *test data* dengan rasio 70:30. Proses ini dilakukan dengan bantuan *train_test_split* dari *library Sklearn*.

Train data merupakan data yang akan digunakan untuk melatih model *deep learning* sebagai contoh model dapat bekerja sesuai dengan parameter yang diberikan. Sedangkan, *test data* merupakan data yang digunakan untuk menguji dan memberikan evaluasi pada model setelah data dilatih.

C. Normalize and Reduce Data

Sub-proses terakhir adalah menormalisasikan serta mereduksi *train data* dan *test data*. Normalisasi data diharapkan dapat meminimalkan penyimpangan data. Pada penelitian ini, normalisasi data dilakukan menggunakan *StandardScaler* dari *library Sklearn*. Selanjutnya melakukan reduksi data dari *train data* dan *test data* hasil normalisasi dengan menggunakan *Principal Component Analysis* (PCA) dari *library Sklearn*. Penentuan jumlah komponen utama dalam PCA, dilakukan dengan menetapkan varians dari data sebesar 95%. Proses reduksi data ini bersifat *optional*.

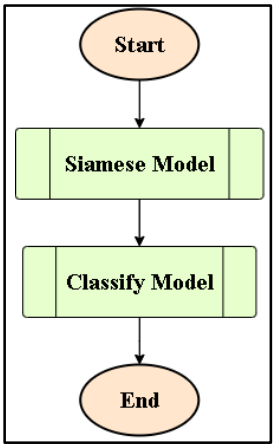


Gambar 3.8 Flowchart Normalize and Reduce Data

3.2.4 Modelling

Pada penelitian ini terdapat 2 model yang digunakan untuk melakukan klasifikasi terhadap perdarahan otak, yaitu dengan menggunakan *Siamese Model* dan *Classify*

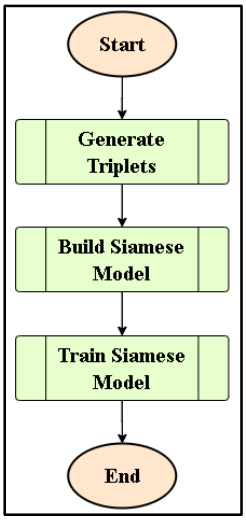
Model. Alur kerja atau *flowchart* dari proses *Modelling* ini pun terbagi menjadi 2 sub-proses, yaitu *flowchart Siamese Model* dan *flowchart Classify Model* yang tergambar pada Gambar 3.9.



Gambar 3.9 Flowchart Modelling

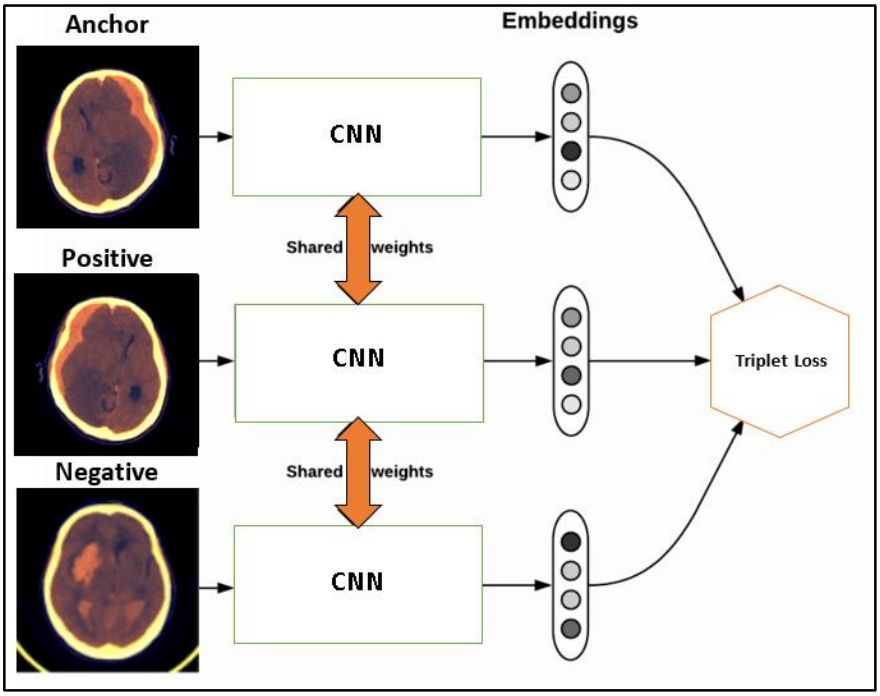
A. *Siamese Model*

Pada sub-proses *Siamese Model* ini, dibangun suatu model yang merupakan salah *satu tipe neurak network*, yaitu *Learning Similarity* dengan *Siamese Neural Network*.



Gambar 3.10 Flowchart Siamese Model

Ide utama dari *Siamese Model* pada penelitian ini adalah melewatkan sebuah input melalui tiga (*triplet*) arsitektur jaringan yang identik yang berbagi nilai parameter yang sama.

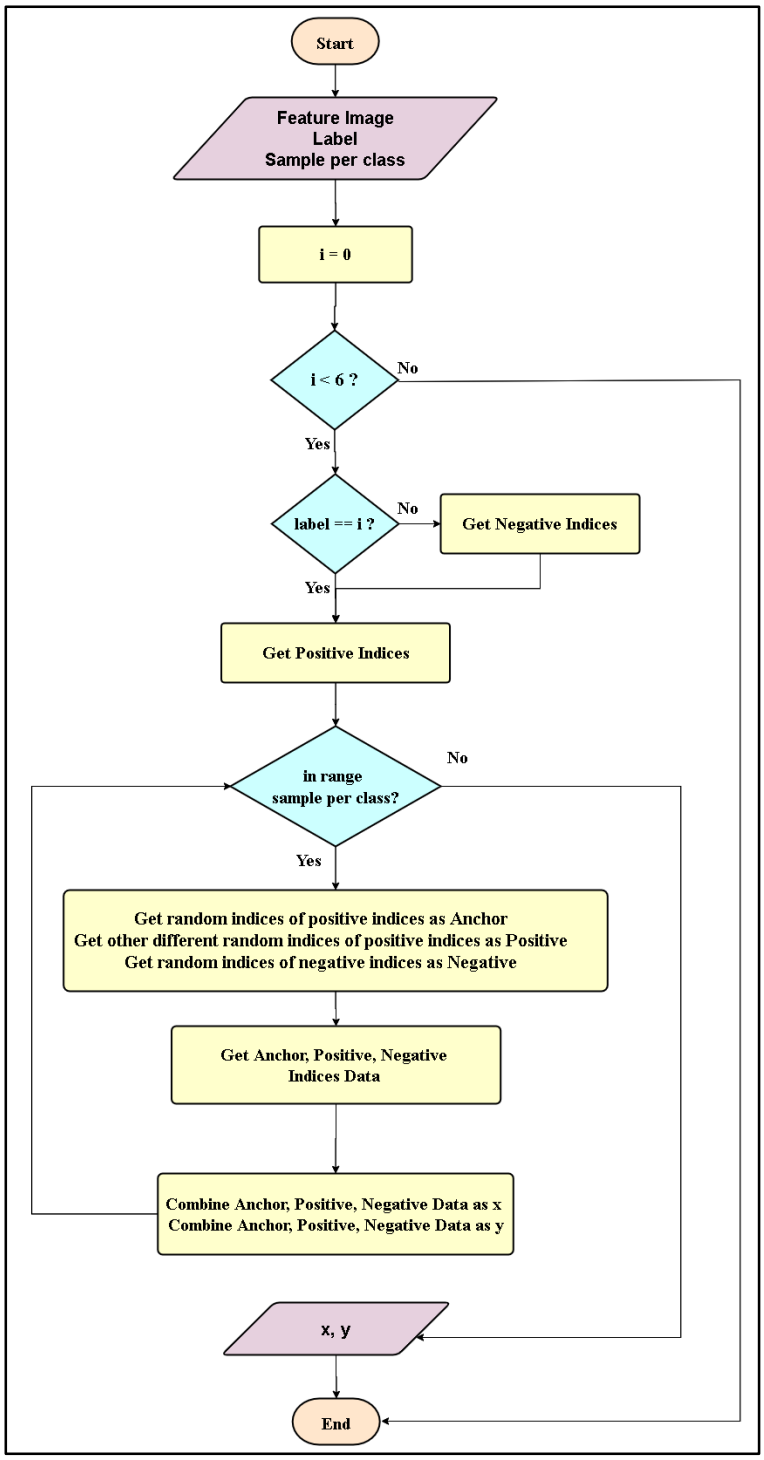


Gambar 3.11 Ilustrasi Siamese Model

Untuk itu, tahap pertama yang perlu dilakukan untuk membangun *Siamese Model* ini adalah *Generate triplet*, kemudian disusul dengan membangun *Siamese Model* dan diakhiri dengan *Train Siamese Model*.

A.1 Generate Triplets

Fungsi *generate triplet* digunakan untuk menghasilkan *input* pasangan citra dalam bentuk *triplet* yang disebut citra *Anchor*, *Positive* dan *Negative*.



Gambar 3.12 Flowchart Generate Triplets

Langkah pertama yang perlu dilakukan adalah menentukan data berisikan fitur citra, label data dan jumlah pasangan data untuk setiap kelas yang pada penelitian ini

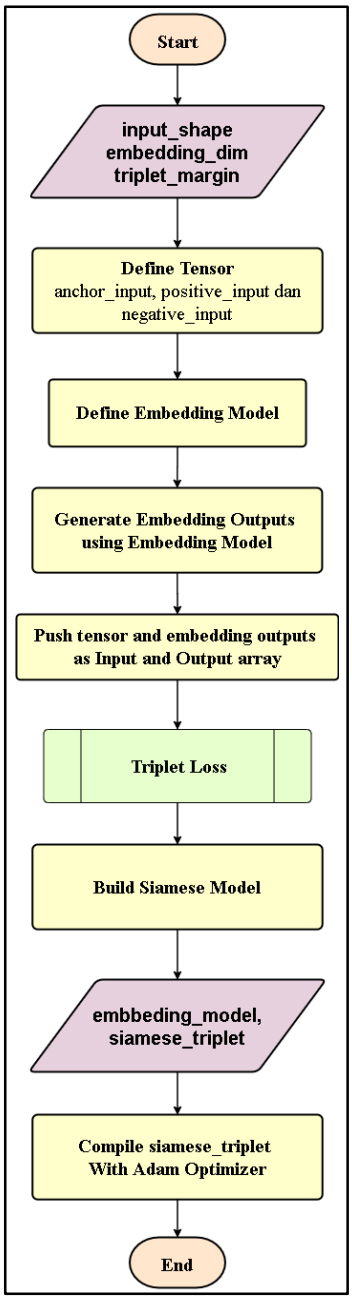
berjumlah 2500. Kemudian dilakukan perulangan sebanyak jumlah kelas, pada setiap perulangan dilakukan pengecekan label dimana ketika terdapat kesamaan label akan terpilih *positive indices* yang merupakan *indices* dari data dengan kelas yang sama, sedangkan jika terdapat perbedaan label, maka akan terpilih *negative indices* yang merupakan *indices*dari data yang memiliki kelas yang berbeda.

Langkah berikutnya, pada setiap perulangan dilakukan pemilihan *indices* untuk data *anchor* dari *positive indices* secara acak, pemilihan *indices* untuk data *positive* dari *positive indices* yang berbeda secara acak dan pemilihan *indices* untuk data *negative* dari *negative indices* secara acak. Setiap pemilihan dilakukan sebanyak jumlah pasangan *triplet* yang telah ditentukan. Hingga langkah ini, telah diperoleh 3 *indices* dari data untuk pasangan *triplet* (*anchor*, *positive* dan *negative*).

Terakhir, dilakukan pengambilan data untuk menjadi pasangan *triplet* berdasarkan *indices* yang telah diperoleh, kemudian dilakukan penggabungan terhadap data fitur *image* dan data label dari pasangan *triplet* tersebut. Fungsi ini akan mengeluarkan *output* gabungan data pasangan *triplet* sebagai x serta label dari data pasangan *triplet* sebagai y.

A.2 Build Siamese Model

Fungsi *build siamese model* digunakan untuk menjabarkan proses pembuatan model *siamese neural network*. Pengembangan flowchart untuk fungsi ini dapat terlihat pada Gambar 3.13.



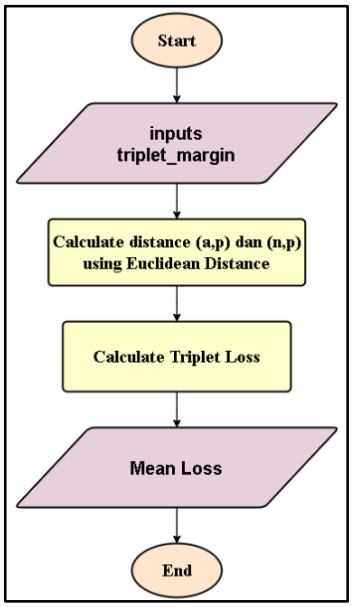
Gambar 3.13 Flowchart Build Siamese Model

Fungsi akan menerima *input_shape* untuk menentukan pasangan *triplet* yang akan menjadi *tensor* dalam *siamese model*. Selain itu, fungsi juga akan menerima *embedding_dim* sebagai dimensi layer *embedding*, dan *triplet_margin* sebagai *margin triplet loss*. Variabel *embedding_model* akan menampung model *hidden layer* yang

terdiri dari 1 *layer* dengan *dropout* sebesar 0,5. Hasil dari *embedding_model* akan digabungkan dan dimasukan ke variabel outputs, sedangkan pasangan *triplet* yang telah ditentukan akan digabungkan dan masuk ke variabel inputs. *Inputs* dan *outputs* ini dihubungkan dalam *siamese model* dengan ditambahkan *loss function* berupa fungsi *triplet loss*. Fungsi *build siamese model* akan mengeluarkan 2 *output* model, yaitu *embedding_model* dan model *siamese_triplet*. Model *siamese_triplet* ini akan di-*compile* dengan *Adam Optimizer* dalam penggunaanya saat proses *training*.

A.2.1 Triplet Loss

Flowchart dari *loss function triplet loss* digambarkan pada Gambar 3.13.



Gambar 3.14 Flowchart Triplet Loss

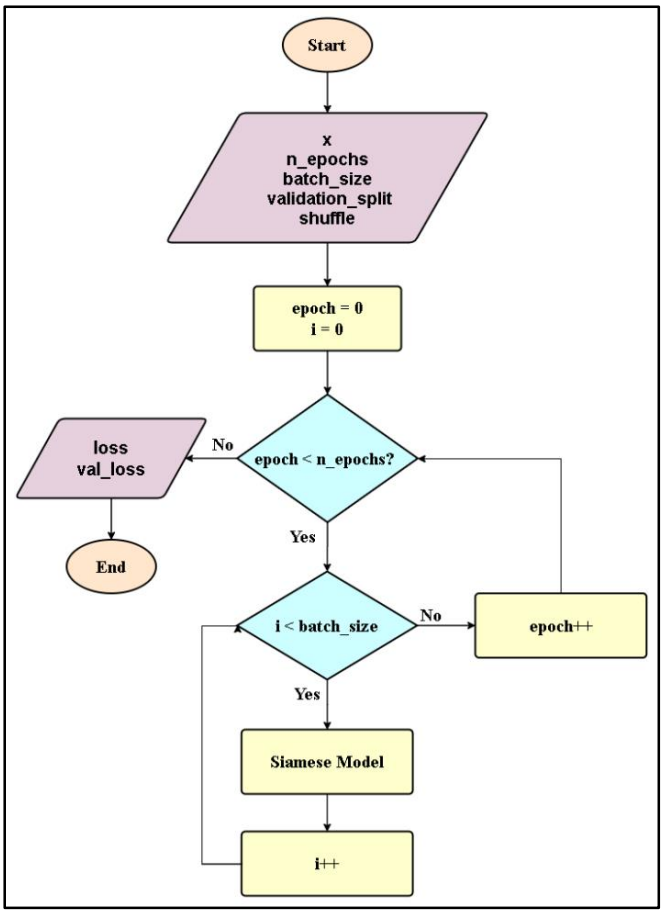
Fungsi ini bertujuan agar jarak antara citra *anchor* dengan citra *positive* menjadi lebih dekat daripada dengan citra *negative*. *Triplet Loss* membutuhkan komponen utama berupa 3 pasangan citra yang disebut pasangan *triplet* yang terdiri atas citra *anchor* sebagai citra acuan, citra *positive* sebagai citra yang memiliki kelas yang sama

dengan citra *anchor* dan citra *negative* yang memiliki kelas yang berbeda dengan citra *anchor*. Selain menerima pasangan *triplet*, fungsi ini juga menerima *triplet_margin* yang bertugas untuk memastikan adanya selisih nilai antara jarak *positive-anchor pair* yang jauh lebih kecil dari *negative-anchor pair* namun tidak melebihi nilai *margin* sehingga nilai *loss* tetap berada pada nilai *positive*. Pada penerapannya, *triplet_margin* bernilai 0,3.

Perhitungan jarak *positive-anchor* dan *negative-anchor* menggunakan *Euclidean Distance*. Keluaran dari perhitungan jarak dengan *Euclidean distance* akan dihitung *loss*-nya menggunakan *Triplet Loss function*. Fungsi *Triplet Loss* akan mengeluarkan *output* berupa nilai *mean* dari hasil perhitungan loss dengan *Triplet Loss*.

A.3 Training Model

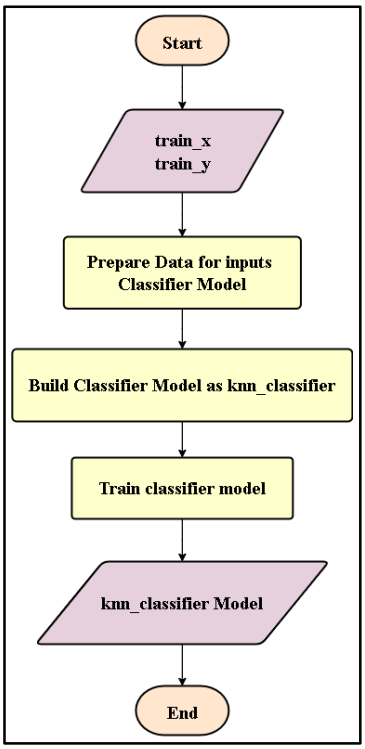
Model yang telah di-*compile* pada fungsi *build siamese model* akan masuk ke subproses *training*. Sub-proses ini akan menerima parameter x sebagai *input training*, *triplet_margin* sebagai margin *triplet loss*, *n_epochs* sebagai jumlah *epoch*, *batch_size* sebagai ukuran data pada setiap *epoch*, *validation_split* sebagai pemecah data *train* menjadi data validasi dan *shuffle* sebagai penentu dilakukan atau tidaknya pengacakan data sejumlah *batch*. Untuk setiap *epoch*, *training* dan *validation* akan dilakukan sejumlah *n_epochs*. Data yang digunakan untuk *training* dan *validation* didapatkan dari hasil *generate triplet* sesuai dengan urutan *batch*. Hasil *training* ini akan menampilkan output *loss* dan *val_loss*. Nilai *loss* didapat dari perhitungan *triplet loss* dengan nilai yang tidak kurang dari nilai *triplet_margin* yang telah diterima. Flowchart fungsi *training* dapat dilihat pada Gambar 3.14.



Gambar 3.15 Flowhcart Training Model

B. *Classify Model*

Flowchart untuk sub-proses *Classify Model* ditunjukkan pada Gambar 3.15. Dalam sub-proses ini, dibangun model untuk melakukan klasifikasi data dengan menggunakan model KNN dengan data yang telah didapat dengan memanfaatkan *embedding_model* yang telah dibangun sebelumnya.

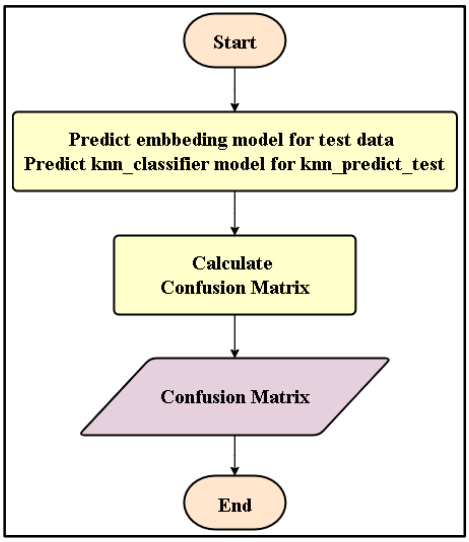


Gambar 3.16 Flowchart Classify Model

Langkah pertama dalam alur kerja sub-proses ini adalah menerima *train_x* sebagai data fitur citra dan *train_y* sebagai kelas citra tersebut. Kemudian akan dilakukan persiapan data sebagai *input* dari *Classify Model* dengan memprediksi *train_embeds* sebagai nilai dari pasangan *triplet train_x* dan menyiapkan variabel target yang berisi data *train_y*.

Pembangunan *Classify Model* menggunakan *class* KNN dari *library Sklearn* dengan menentukan nilai K sebagai tetangga dan *distance metric* yang digunakan. Pada penelitian ini digunakan nilai K sebesar 7 dan *euclidean* sebagai *distance metric*. Model yang telah dibuat ini berikutnya akan di-*train* dengan data *train_embeds* dan target yang telah disiapkan sebelumnya dan mengeluarkan *output* berupa *knn_classifier* sebagai *Classify Model*.

3.2.5 Evaluation Result



Gambar 3.17 Flowchart Evaluation Result

Pada Gambar 3.16, dapat dilihat *flowchart* sub-proses *evaluation result*. Sub-proses ini berfungsi untuk memprediksi data *testing*. Variabel *knn_predict_test* akan berisi hasil dari prediksi data *testing* pada *embedding model* yang disebut dengan *embedding space*. *Embedding space* berikutnya akan diprediksi menggunakan model *knn_classifier* untuk mendapatkan nilai kelas dari hasil prediksi model. Nilai kelas data *testing* dan nilai kelas hasil dari prediksi model akan digunakan untuk perhitungan metrik-metrik yang dalam melakukan evaluasi model yang telah di-*training*. Perhitungan ini dilakukan dengan menggunakan *confusion matrix*, yang nantinya akan menampilkan *confusion matrix*, *accuracy*, *f1-score*, *precision score*, *recall score*.