BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Dalam penelitian yang dilakukan, tahap penelitian akan dibagi menjadi 5 tahap :

1. Studi Literatur
   Untuk dapat memulai penelitian, maka akan diperlukan tahap mencari dan membaca berbagai literatur. Diharapkan melalui tahapan ini, terjadi peningkatan pemahaman mengenai *siamese convolutional neural network*, *triplet loss*, MaskTheFace *tools*, arsitektur FaceNet, arsitektur ArcFace, dan arsitektur OpenFace.

2. Pengumpulan Dataset
   Dalam tahapan kedua, akan dikumpulkan dataset-dataset yang akan dibutuhkan dalam tahap perancangan dan implementasi model. Dataset yang digunakan adalah sampel dari VGG Face2 Dataset yang berisi kumpulan gambar wajah. Dataset tersebut akan ditingkatkan dengan memanfaatkan MaskTheFace tools untuk memberikan gambar masker pada wajah-wajah yang ada dalam dataset tersebut.

3. Perancangan dan Implementasi Model
   Pada tahap ini, akan dilakukan implementasi model dengan bantuan literatur-literatur yang telah diperoleh. Model akan dibuat dalam bentuk *network* yang akan menerima tiga buah *embedding* sebagai input. Untuk membangun *embedding*, akan dimanfaatkan *pre-trained model*. Penjelasan mengenai perancangan dan implementasi model akan dijelaskan dalam bagian 3.2 dan 4.3.

4. Pengujian dan Perbaikan Model
   Setelah perancangan dan implementasi, maka setiap performa dari *pre-trained model* akan diuji berdasarkan metriks *Max Accuracy*, *False Acceptance Rate*, dan *False Rejection Rate*. Ketika terjadi hasil yang masih kurang berkenan dari setiap model tersebut, akan dilakukan *fine-tuning* parameter-parameter yang digunakan selama proses pelatihan model atau mengolah kembali dataset yang digunakan agar dapat mendapatkan hasil yang optimal untuk seluruh arsitektur yang digunakan.

5. Penulisan Laporan

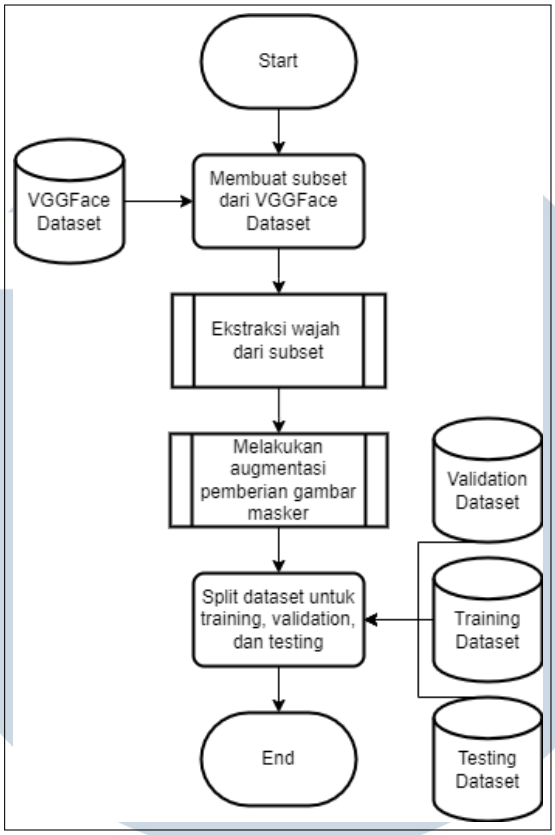
   Pada tahap terakhir ketika aplikasi sudah berhasil dibangun dan mendapatkan hasil yang diinginkan, laporan akan disusun sebagai dokumentasi dari model yang telah dibangun. Penulisan laporan akan dilakukan sesuai dengan format yang dibutuhkan.

3.2 Perancangan Model

Perancangan program untuk mengimplementasikan *siamese convolutional neural network* dalam *masked face recognition* dapat dibagi menjadi 3 bagian utama. Ketiga bagian tersebut adalah *generate dataset*, *training model*, dan *evaluate model*.

1. Generate Dataset

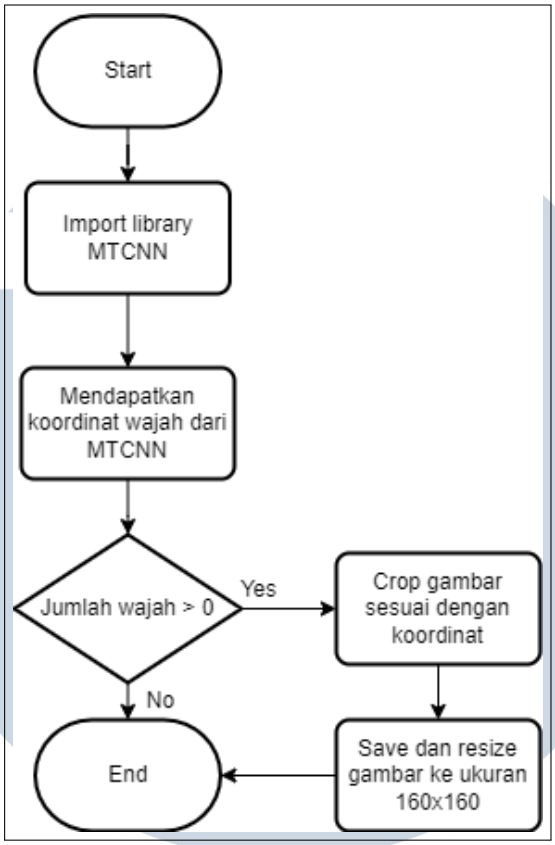
   Sesuai dengan gambar 3.1, akan dibuat subset dari VGG Face2 Dataset. Subset yang akan digunakan dalam proses pelatihan adalah sebesar 1500 gambar. Jumlah kelas dalam subset yang diambil berjumlah 15 dengan 100 gambar per kelasnya. Untuk dapat mendapatkan hasil yang optimal ketika menjalankan pelatihan, maka akan dilakukan ekstraksi wajah pada subset dataset yang digunakan. Hal ini dilakukan untuk mengurangi *noise* yang ada dalam gambar dan membantu model untuk dapat membedakan bagian wajah pada gambar. Setelah dilakukan ekstraksi wajah, dataset tersebut akan ditingkatkan dengan memanfaatkan MaskTheFace *tools* untuk memberikan gambar masker pada wajah-wajah yang ada dalam dataset tersebut. Melalui augmentasi yang dilakukan, total dataset dari 1500 gambar akan meningkat menjadi 3000 gambar. Augmentasi pemberian gambar masker akan memberikan masker medis berwarna putih. Hasil ekstraksi serta augmentasi tersebut lalu akan dibagi menjadi dataset untuk *training*, *validation*, dan *testing*. Sebanyak 2100 gambar akan digunakan sebagai dataset training, dengan 140 gambar per kelas. *Validation* dan *testing* akan masing-masing menggunakan 450 gambar dengan 30 gambar per kelasnya.



Gambar 3.1. Flowchart dari proses *generate dataset*

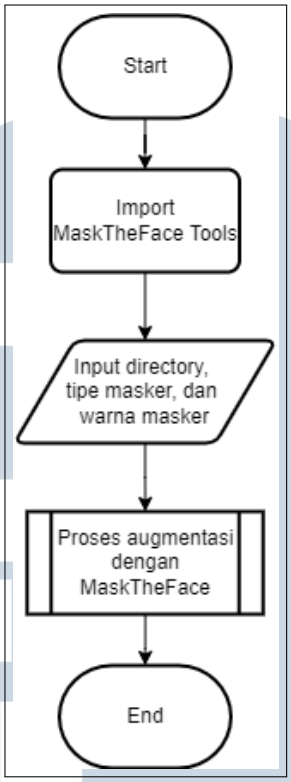
- Ekstraksi Wajah dari Subset

  Proses ekstraksi wajah akan memanfaatkan bantuan library MTCNN. MTCNN atau *Multi-Task Cascaded Convolutional Neural Networks* merupakan salah satu implementasi *face detection* pada Tensorflow. MTCNN dapat mengembalikan koordinat wajah yang ada dalam gambar sesuai dengan jumlah wajah yang terdeteksi. Dataset akan di-*crop* sesuai dengan koordinat yang diperoleh dan di-*resize* ke ukuran 160x160.



Gambar 3.2. Flowchart dari proses ekstraksi wajah dari subset

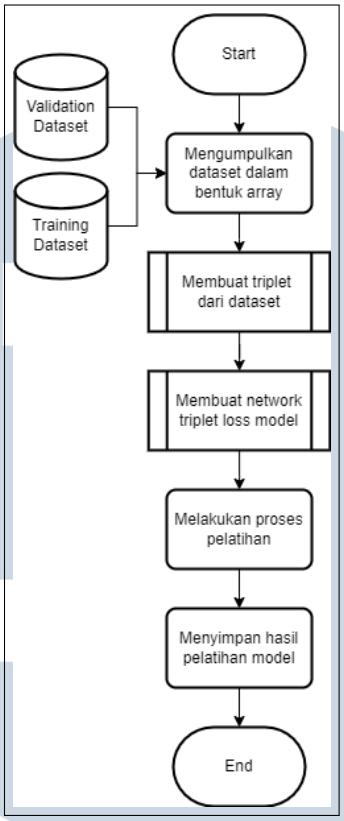
- Melakukan Augmentasi Pemberian Gambar Masker pada Wajah
  Untuk dapat melakukan augmentasi pemberian gambar masker pada dataset, akan dimanfaatkan *tools* MaskTheFace. Sesuai dengan flowchart di gambar 3.3, proses augmentasi akan diawali dengan import *tools* MaskTheFace. Setelah itu, akan diberikan input berupa *directory*, tipe masker, dan warna masker. Berdasarkan input yang diberikan, akan dijalankan proses augmentasi dengan MaskTheFace sesuai dengan alur pada gambar 2.3.



Gambar 3.3. Flowchart dari proses tools MaskTheFace

2. Training Model

   Setelah dataset telah berhasil di-*generate*, proses selanjutnya adalah melakukan pelatihan model. Dapat diketahui dalam flowchart pada gambar 3.4, proses *training* diawali dengan mengumpulkan dataset ke dalam *array*. Berdasarkan kumpulan dataset tersebut, akan dibuat data *triplet* yang terdiri dari gambar *anchor*, gambar *positive* dan gambar *negative*. Triplet yang dibuat akan digunakan sebagai input dari model yang akan dilatih. Tahap selanjutnya adalah membuat model *network triplet loss*. Model tersebut akan dibangun dengan menggunakan tiga buah embedding model yang sama, di mana embedding tersebut akan menggunakan *pre-trained* model *ArcFace*, *Facenet*, atau *Openface*. Model *network triplet loss* tersebut lalu akan dilatih dengan parameter dan hyperparameter yang ditentukan. Metrik dari hasil pelatihan lalu akan disimpan sebagai indikator dari kinerja model setelah dilakukan pelatihan.



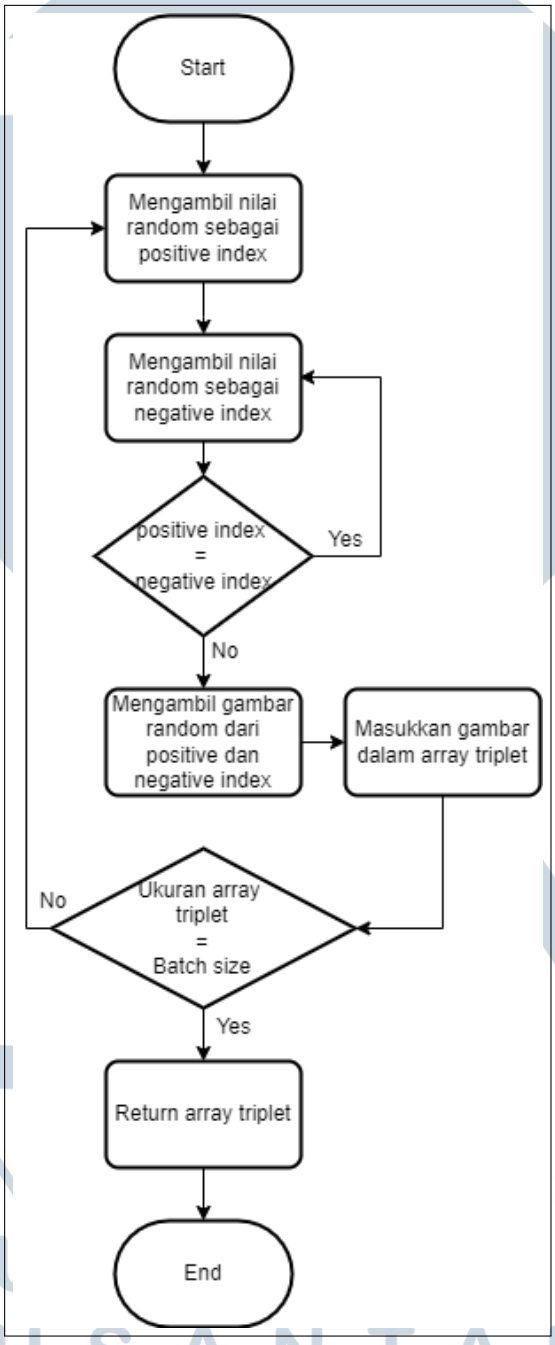
Gambar 3.4. Flowchart dari proses training model

- Membuat Triplet dari Dataset

  Network yang dibuat akan menggunakan fungsi *triplet loss* sebagai output model. Metode *triplet loss* akan menerima tiga buah input. Tiga buah input tersebut adalah gambar *anchor*, gambar *positive*, dan gambar *negative*. Gambar *positive* merupakan gambar lainnya yang berasal dari kelas yang sama dari gambar *anchor*. Sedangkan, gambar *negative* adalah gambar lainnya yang berasal dari kelas berbeda dari gambar *anchor*.

  Pada gambar 3.5, dapat diketahui bahwa proses *generate* triplet dimulai dengan mengambil nilai secara acak untuk dijadikan index dari gambar *positive*. Index *positive* juga akan digunakan sebagai index dari gambar *anchor*. Selanjutnya akan diambil juga nilai secara acak sebagai index untuk gambar *negative*. Proses pengambilan nilai tersebut secara acak akan dilakukan terus menerus jika nilai index gambar *negative* masih sebesar nilai index gambar *positive*. Setelah index gambar *anchor*, *positive*, dan *negative* diterima, maka akan diambil satu gambar secara acak

dari index-index tersebut. Gambar acak tersebut lalu akan dimasukkan ke dalam array triplet. Alur ini akan dilakukan secara terus menerus sampai ukuran triplet sesuai dengan ukuran batch.

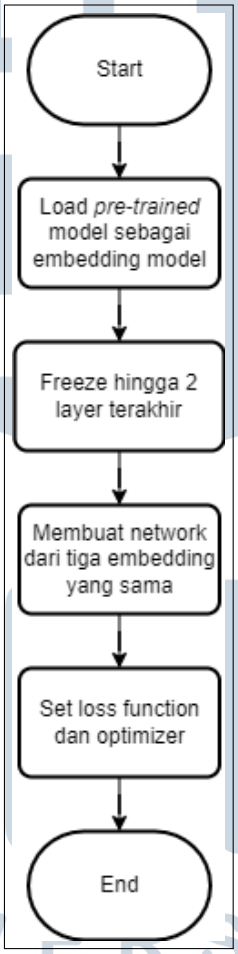


Gambar 3.5. Flowchart dari *generate triplet*

- Membuat Network Triplet Loss Model

  Untuk dapat mengimplementasikan *siamese convolutional neural net-*

work, maka akan dibuat suatu network yang memiliki tiga buah embedding yang sama. Embedding tersebut berasal dari *pre-trained* model yang akan di-*load*. *Pre-trained* model tersebut akan di *freeze* hingga 2 layer terakhir untuk tidak mengubah *convolutional layer* dari model tersebut. Setelah itu sesuai gambar 3.6, akan di-*set* juga *loss function* serta *optimizer* untuk network tersebut. *Loss function* yang digunakan adalah *triplet loss function* dan *optimizer* yang digunakan adalah optimizer SGD atau ADAM.

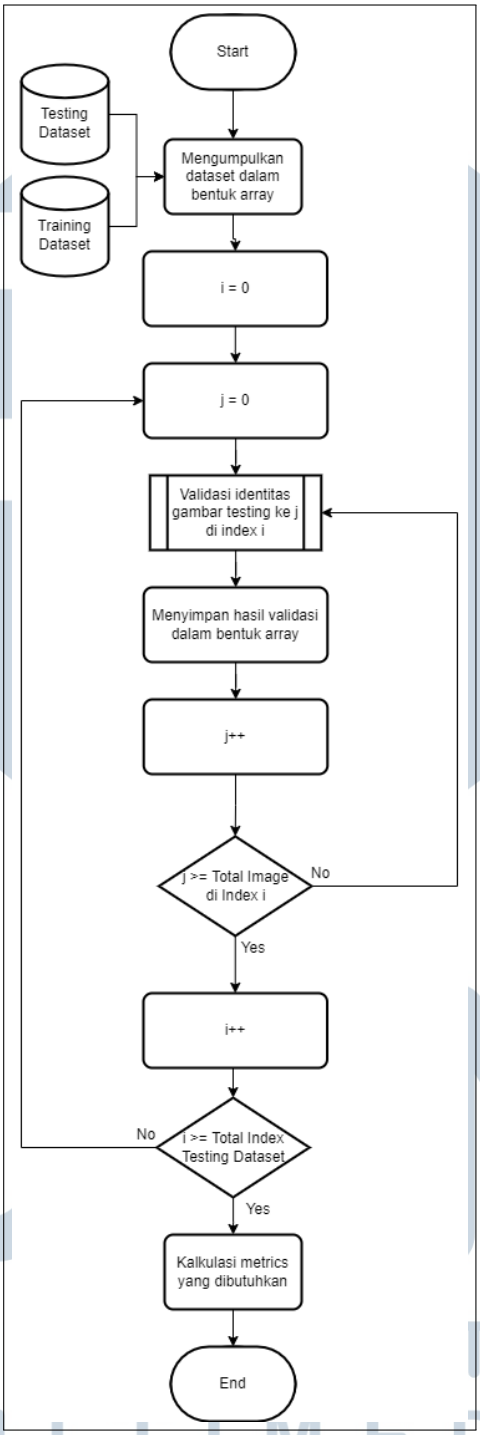


Gambar 3.6. Flowchart dari membuat network triplet loss model

3. Evaluasi Model

   Setelah pelatihan model selesai dilakukan, tahap selanjutnya adalah melakukan evaluasi terhadap model yang telah dibangun. Tahap ini dilakukan untuk memperoleh nilai metriks yang dibutuhkan untuk penilaian model. Pada gambar 3.7, gambar-gambar dataset akan dikumpulkan ke dalam su-

atu array untuk digunakan dalam proses evaluasi. Validasi akan dilakukan untuk mengetahui hasil prediksi index gambar dataset testing. Validasi ini dilakukan dengan membandingkan *distance* dari gambar *testing* dengan sampel gambar yang ada di *training* dataset. Index yang memiliki hasil rata-rata perbandingan *distance* terendah, akan dianggap sebagai index hasil prediksi model. Proses ini dilakukan berulang-ulang sampai seluruh data di dalam *testing* dataset sudah ditentukan hasil validasinya. Hasil validasi tersebut akan diolah menggunakan *library* Sci-Kit Learn untuk dapat memperoleh metriks-metriks yang dibutuhkan.



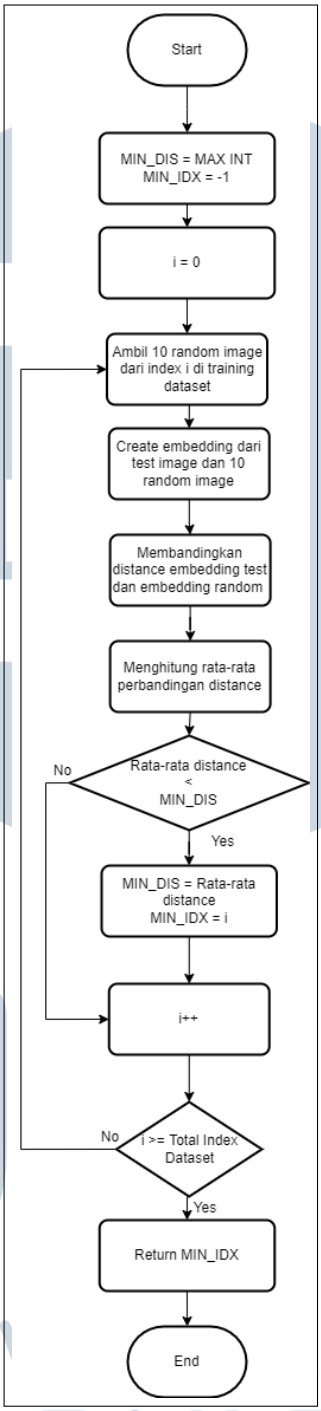
Gambar 3.7. Flowchart dari tahap evaluasi model

- Validasi Identitas

  Tahap validasi dilakukan untuk mengetahui hasil prediksi index *training data* yang memiliki rata-rata *distance* terendah. *Distance* tersebut dapat diperoleh dengan membandingkan hasil embedding gambar *an-*

chor dengan gambar yang ada dalam suatu index *training data*. Untuk mendapatkan hasil tersebut akan dilakukan proses yang dapat dilihat di gambar 3.8.

Dalam gambar 3.8, dapat diketahui bahwa gambar *anchor* akan dibandingkan dengan seluruh kelas yang ada di dalam data *training*. Untuk seluruh kelas yang ada, akan diambil sampel gambar sebanyak 10 secara acak. Setelah itu, akan dibuat *embedding* dari gambar *anchor* dan gambar acak tersebut. Hasil perbandingan kedua embedding tersebut lalu akan digunakan untuk memperoleh rata-rata *distance* perbedaan gambar *anchor* dengan gambar-gambar dalam suatu index. Jika rata-rata *distance* lebih rendah dibandingkan dengan rata-rata *distance* lainnya, maka index terendah adalah index *training* data tersebut.



Gambar 3.8. Flowchart dari tahap validasi identitas