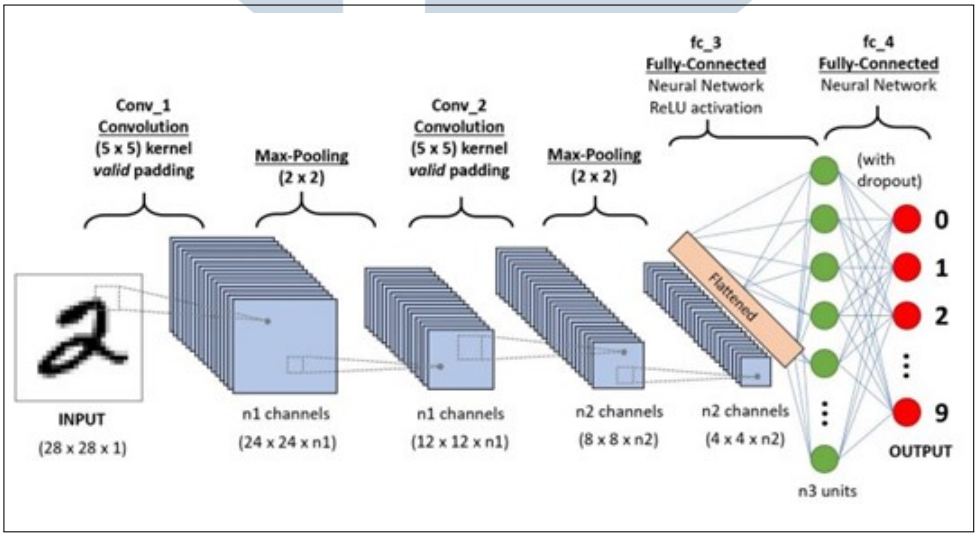
BAB 2
LANDASAN TEORI

Untuk dapat mengimplementasikan siamese convolutional neural network untuk masked face recognition, maka akan dimanfaatkan beberapa teori-teori yang mendukung penelitian yang sedang dilakukan. Teori yang akan digunakan adalah sebagai berikut:

2.1 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan algoritma *Deep Learning* yang kerap digunakan dalam *Image Processing* dan *Image Classification* [9]. Struktur dari suatu CNN dapat dilihat pada gambar 2.1. CNN menerima input berupa gambar dengan bias serta bobot pembelajaran untuk dapat membedakan suatu gambar dengan gambar lainnya.



Gambar 2.1. *Sequence* CNN dalam Melakukan Klasifikasi
Sumber: [9]

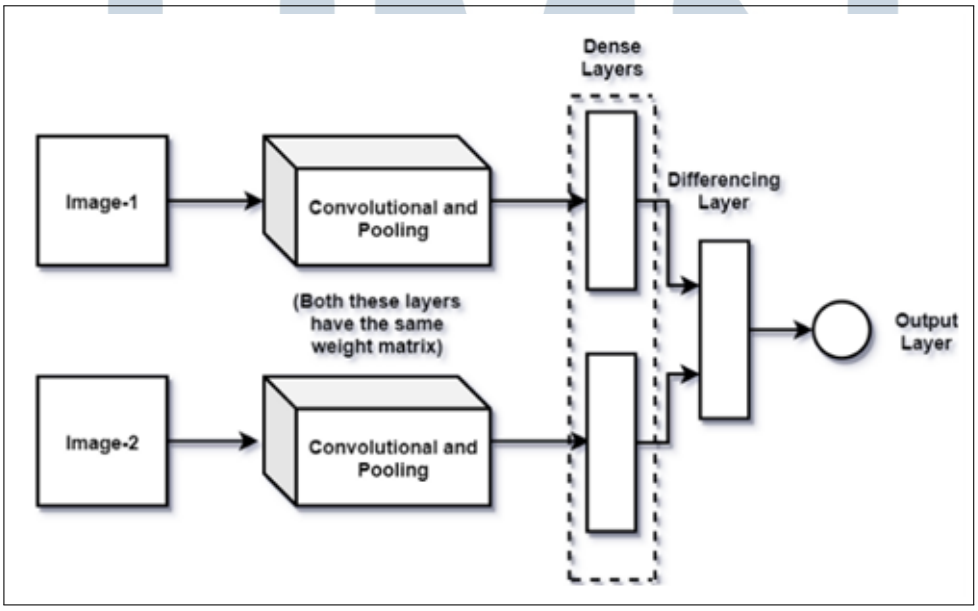
Suatu CNN dibagi menjadi beberapa bagian, bagian pertama dari CNN Bernama Kernel [9]. Kernel berfungsi untuk menerima input gambar dalam bentuk matriks. Bagian ini akan membantu CNN untuk mengekstraks fitur yang diperoleh dari suatu gambar. Setelah melewati kernel, gambar akan masuk ke dalam *convolutional layer* dan *pooling layer*. Bagian ini penting untuk mengekstrasi fitur-fitur dominan yang ada di dalam suatu gambar. Tipe *pooling* yang kerap digunakan

adalah *Max Pooling* untuk mendapatkan nilai maksimal bagian gambar yang telah diterima dari kernel dan *Average Pooling* untuk mendapatkan nilai rata-rata bagian gambar yang telah diterima oleh kernel [9].

Fully-Connected Layer akan menerima ekstraksi input dari pooling dan akan mengubah gambar menjadi suatu vektor [9]. Vektor tersebut yang akan digunakan dalam *neural network* dan *backpropagation* akan dilakukan untuk setiap iterasi pelatihan. Setelah menjalani beberapa iterasi (epochs), model CNN akan dapat membedakan fitur-fitur dominan yang ada dalam suatu gambar dan mengklasifikasikan gambar tersebut.

2.2 Siamese Convolutional Neural Network dan Triplet Loss

Siamese neural network adalah suatu jaringan yang terdiri dari beberapa network kembar yang menerima input berbeda tapi tergabung dalam suatu *function top-level* [10]. Fungsi ini menghitung beberapa metrik antara representasi fitur di setiap sisi network. Network yang berhubungan tersebut akan memiliki *weight* yang sama sehingga menjamin input yang sama tidak mungkin dapat dipetakan ke lokasi yang berbeda. *Siamese neural network* kerap digunakan bersama dengan *convolutional neural network* karena dapat mencapai hasil yang baik dalam tugas pengenalan gambar. Contoh struktur *siamese convolutional neural network* dengan dua network kembar dapat dilihat pada gambar 2.2.



Gambar 2.2. Struktur Siamese Convolutional Neural Network

Sumber: [11]

Salah satu metode yang kerap digunakan dalam *siamese neural network* adalah metode bernama *triplet loss*. Metode tersebut memanfaatkan *siamese neural network* dengan tiga subnetwork yang identik. Tiga gambar akan diberikan sebagai input model, di mana dua di antaranya serupa (*anchor* dan sampel positif), dan yang ketiga tidak serupa (sampel negatif). Tujuan diberikannya ketiga gambar tersebut adalah agar model dapat belajar memperkirakan kesamaan antara gambar sampel positif dan sampel negatif.

Metode triplet loss akan membuat suatu embedding dari gambar dalam suatu eucladian space [4]. Dalam metode ini, akan diharapkan bahwa gambar anchor akan memiliki jarak yang lebih mendekati gambar-gambar dalam sampel positif dibandingkan dengan gambar-gambar sampel negatif.

$$\| f(x_i^a) - f(x_i^p) \|_2^2 + \alpha < \| f(x_i^a) - f(x_i^n) \|_2^2, \tag{2.1}$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \tau \tag{2.2}$$

$$L = \sum_i^N \left[\| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + \alpha \right] \tag{2.3}$$

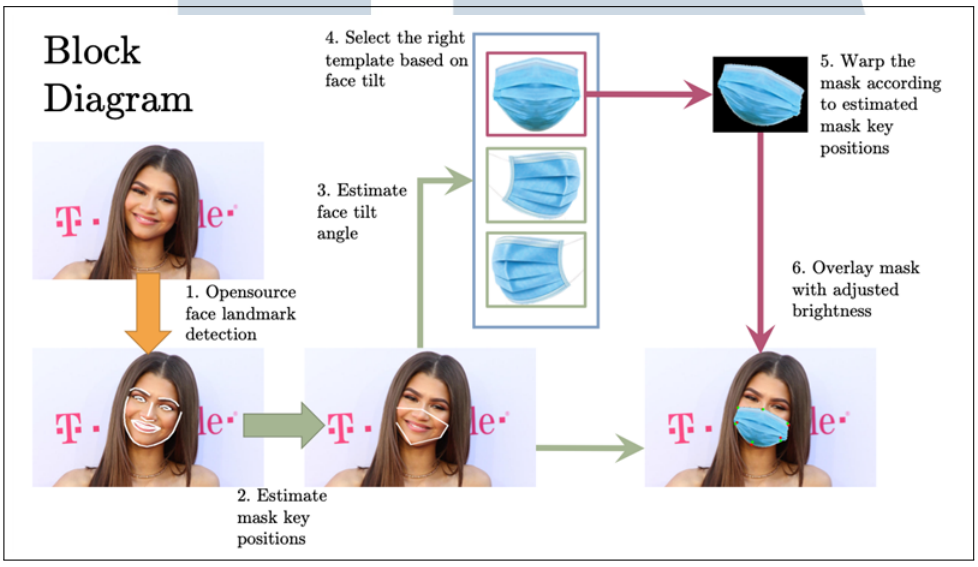
Berikut adalah arti dari simbol-simbol yang digunakan:

- x_i^a : gambar anchor
- x_i^p : gambar sampel positif
- x_i^n : gambar sampel negatif
- α : margin antara sampel positif dan sampel negatif
- τ : kumpulan seluruh *triplet* yang memiliki kardinalitas N
- N : kardinalitas
- L : nilai loss function

2.3 MaskTheFace

MaskTheFace merupakan suatu *open source tools* yang dapat digunakan untuk memberikan masker pada input gambar wajah. *Tools* tersebut merupakan suatu

script berbasis *computer vision* yang memanfaatkan dlib yang mengidentifikasi fitur wajah dan akan memberikan masker pada bagian mulut dan hidung ke input wajah. Berdasarkan kemiringan wajah, template masker akan dipilih dari *library* gambar masker agar dapat memberikan bentuk masker yang cocok pada gambar wajah. MaskTheFace memiliki 5 tipe masker dengan berbagai variasi tekstur warna dan dapat menerima input satu gambar atau satu directory berisi gambar. Alur kerja dari MaskTheFace dapat dilihat pada gambar 2.3.

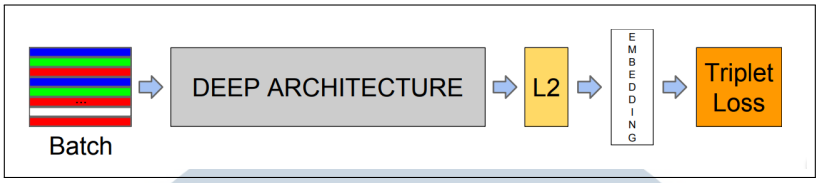


Gambar 2.3. Diagram Cara Kerja *Tools* MaskTheFace

Sumber: [6]

2.4 FaceNet

FaceNet merupakan model yang menggunakan *deep convolutional network* untuk melakukan verifikasi wajah. Untuk dapat membangun model tersebut, FaceNet akan melakukan triplet loss untuk melakukan *face verification*, *face recognition* dan *face clustering* [4]. Sesuai dengan gambar 2.4, FaceNet akan membuat suatu embedding dari gambar yang ada dalam suatu *batch* dan embedding tersebut akan digunakan dalam fungsi triplet loss. Hal tersebut dilakukan untuk mendapatkan fitur-fitur dari gambar wajah dalam kondisi-kondisi dan bentuk-bentuk berbeda.

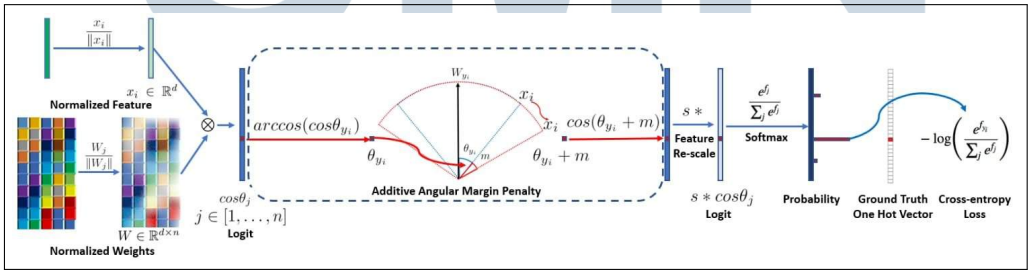


Gambar 2.4. Struktur Model FaceNet
Sumber: [4]

FaceNet menggunakan sekitar 100-200 juta gambar wajah untuk pelatihan yang terdiri dari sekitar 8 juta identitas berbeda. Detektor wajah digunakan pada setiap gambar untuk membentuk *bounding box* di sekitar setiap wajah. Gambar wajah tersebut diubah ukurannya ke ukuran input dari jaringan masing-masing. Ukuran input yang akan digunakan sekitar 96x96 pixel hingga 224x224 pixel.

2.5 ArcFace

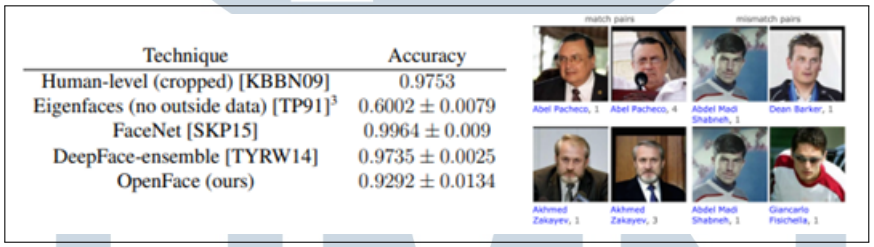
ArcFace adalah model *machine learning* yang mengambil dua gambar wajah sebagai input dan mengeluarkan jarak di antara mereka untuk melihat seberapa besar kemungkinan mereka adalah orang yang sama [3]. ArcFace kerap digunakan untuk pengenalan wajah dan pencarian wajah. ArcFace menggunakan mekanisme *similarity learning* yang memungkinkan dilakukannya klasifikasi dengan memperkenalkan metode *Angular Margin Loss*. Sesuai dengan gambar 2.5, jarak antar wajah akan dihitung menggunakan *cosine similarity* dengan menambahkan margin pada sudut yang digunakan dalam perhitungan. Setelah perhitungan dilakukan, lapisan *Fully Connected (FC)* mengambil inner product dari fitur serta bobot dan menerapkan *Softmax* ke output.



Gambar 2.5. Struktur Model ArcFace
Sumber: [3]

2.6 OpenFace

OpenFace merupakan suatu model *face recognition* yang dibangun berdasarkan inspirasi dari model DeepFace oleh Facebook dan FaceNet oleh Google [5]. OpenFace dibangun dengan tujuan utama untuk membuat suatu model yang dapat digunakan dengan mudah dalam suatu aplikasi *mobile* secara *real-time*. OpenFace akan melakukan pre-process wajah dan akan dikirim sebagai input ke dalam *neural network*, di mana akan dilakukan ekstraksi fitur-fitur utama dari setiap wajah masing-masing orang. OpenFace menggunakan metode triplet loss yang digunakan oleh model FaceNet. OpenFace memanfaatkan dataset berukuran 500.000 gambar dari menggabungkan dua dataset face recognition berlabel terbesar, CASIA-WebFace [YLLL14] dan FaceScrub [NW14]. Ukuran dataset yang digunakan jauh lebih kecil dibandingkan dengan FaceNet yang menggunakan 100-200 juta gambar dan DeepFace yang menggunakan 4.4 juta gambar. Pada gambar 2.6, dapat diketahui bahwa OpenFace tetap memperoleh akurasi yang sebanding dengan kedua model tersebut walaupun menggunakan dataset yang berukuran jauh lebih kecil.



Technique	Accuracy
Human-level (cropped) [KBBN09]	0.9753
Eigenfaces (no outside data) [TP91][3]	0.6002 ± 0.0079
FaceNet [SKP15]	0.9964 ± 0.009
DeepFace-ensemble [TYRW14]	0.9735 ± 0.0025
OpenFace (ours)	0.9292 ± 0.0134

Gambar 2.6. Akurasi OpenFace Dibandingkan Model Lainnya
Sumber: [5]

2.7 Metriks Evaluasi

Untuk melakukan evaluasi terhadap model yang telah dibuat, maka akan digunakan metrik *Accuracy*, *False Acceptance Rate (FAR)*, dan *False Rejection Rate (FRR)* yang merupakan salah satu metriks dalam arsitektur FaceNet [16].

- *Accuracy*
  Accuracy menggambarkan seberapa akurat model dalam mengidentifikasi pasangan gambar yang diberikan. Nilai akurasi dapat dihitung dengan persamaan :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

- *False Acceptance Rate (FAR)*
  FAR merupakan metrik yang digunakan untuk mengukur rasio diterimanya pengguna yang tidak sesuai [8]. Nilai FAR dapat dihitung dengan persamaan :

$$FAR = \frac{FP}{FP + TN} \tag{2.5}$$

- *False Rejection Rate (FRR)*
  FRR merupakan metrik yang digunakan untuk mengukur rasio ditolaknya pengguna yang sesuai [8]. Nilai FRR dapat dihitung dengan persamaan :

$$FRR = \frac{FN}{FN + TP} \tag{2.6}$$

  Berikut adalah arti dari simbol-simbol yang digunakan:

  - *TP* : *True Positive*
  - *FP* : *False Positive*
  - *TN* : *True Negative*
  - *FN* : *False Negative*

2.8 Tensorflow dan Keras

TensorFlow adalah suatu *interface* untuk mengekspresikan dan mengimplementasi algoritma pembelajaran mesin [12]. Komputasi yang diekspresikan menggunakan TensorFlow dapat dijalankan dengan sedikit atau tanpa perubahan pada berbagai macam sistem, mulai dari *mobile devices* hingga sistem terdistribusi skala besar yang terdiri dari ratusan mesin dan ribuan perangkat komputasi seperti kartu GPU. Sistem Tensorflow dapat digunakan untuk melakukan pelatihan dan inferensi

model *neural network*. Selain itu, Tensorflow juga telah digunakan untuk penelitian dan menerapkan sistem pembelajaran mesin ke dalam berbagai bidang ilmu computer seperti pengenalan suara, *computer vision*, robotika, pencarian informasi, pemrosesan bahasa alami, dan berbagai hal lainnya.

Keras adalah suatu API dalam bahasa Python yang dibangun di atas platform Tensorflow untuk melakukan *deep learning* [13]. Keras memudahkan penggunanya untuk dapat mengatur *layer network* dengan memanfaatkan API yang bersifat *high-level*. Jika penelitian tidak merujuk pada struktur *neural network* yang sepenuhnya baru, kecil kemungkinan perlu untuk memprogram TensorFlow secara langsung tanpa memanfaatkan keras.

2.9 Optimizer Adam dan SGD

Optimizer Adaptive Moment Estimation atau Adam adalah sebuah algoritma optimisasi berbasis gradien pada fungsi *stochastic* [14]. Metode Adam menghitung secara adaptif *learning rate* untuk parameter yang berbeda dari perkiraan momen pertama dan kedua dari gradien; nama Adam berasal dari estimasi momen yang adaptif. Metode ini mudah diterapkan, efisien secara komputasi, memiliki sedikit kebutuhan memori, tidak berubah terhadap penskalaan ulang gradien, dan sangat cocok untuk masalah yang besar dalam hal data dan/atau parameter.

Optimizer *Stochastic Gradient Descent* atau SGD adalah algoritma optimisasi yang bertujuan untuk meminimalkan risiko empiris model dengan menghitung berulang kali gradien dari *loss function* pada salah satu contoh pelatihan, atau beberapa contoh *batch*, dan memperbarui parameter model [15]. SGD bersifat *scallable*, *robust*, dan berkinerja baik di banyak domain berbeda mulai dari masalah yang mulus dan sangat cembung hingga tujuan non-cembung yang kompleks.