



Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

BAB III

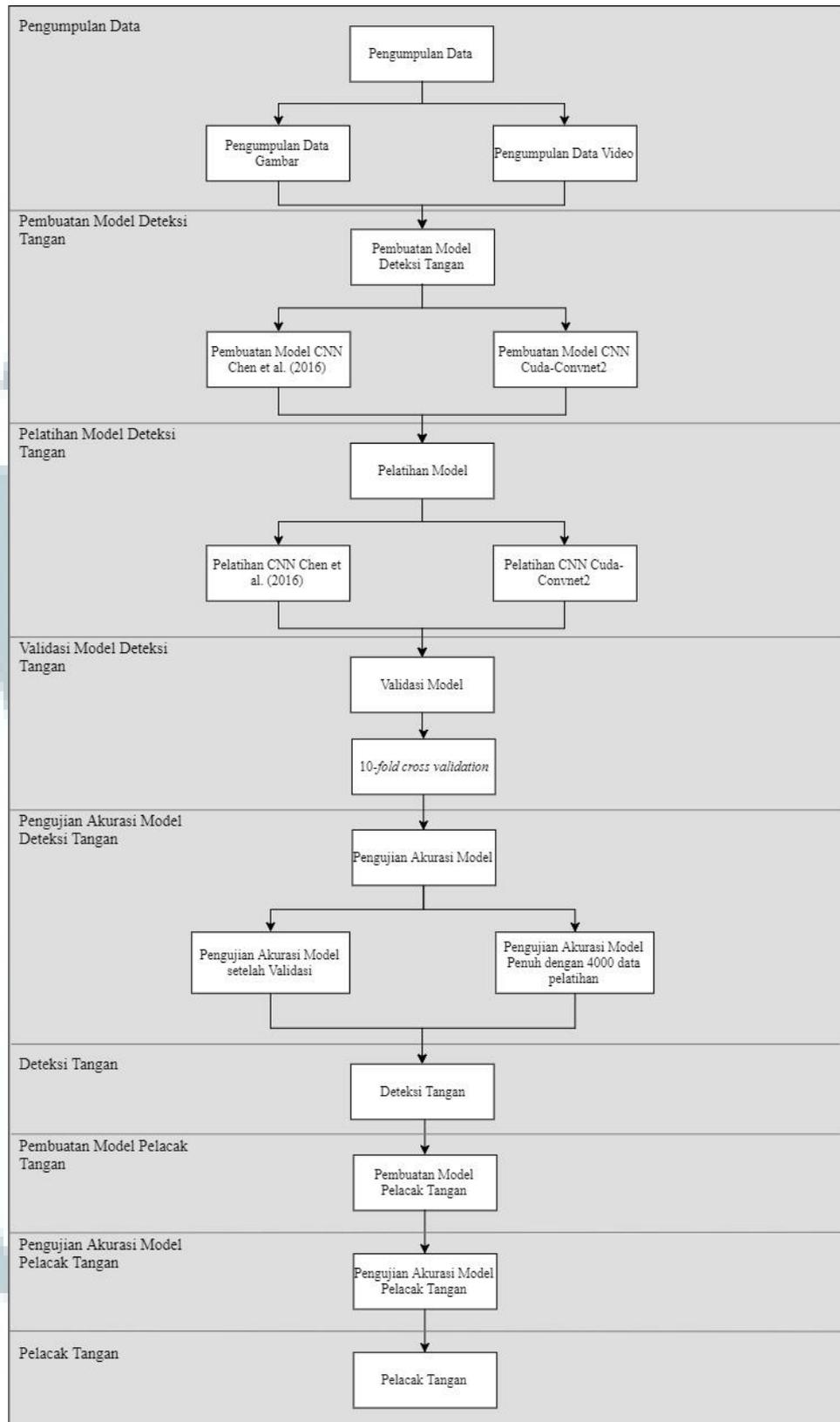
METODE PENELITIAN

3.1 Gambaran Umum

Penelitian ini bertujuan untuk mendeteksi dan melacak tangan pada gambar dimana menggunakan *deep learning* sebagai alat untuk mendeteksi dan melacak tangan sebagai objek dalam gambar. Deteksi tangan pada penelitian ini menggunakan gambar statis dimana gambar tersebut dibagi menjadi dua jenis yaitu gambar tangan dan gambar bukan tangan. Gambar tangan harus memiliki kondisi dimana tidak ada objek lain selain tangan dan latar yang kompleks dan tangan dapat memiliki bentuk yang bervariasi, dan gambar bukan tangan bisa gambar yang merupakan objek selain bukan tangan. Pelacak tangan pada penelitian ini menggunakan gambar kontinu (video) sebagai bentuk data. Kondisi tangan dapat bervariasi dan dapat disertai objek lain serta memiliki latar yang kompleks. Video akan digunakan untuk mendeteksi dan melacak letak tangan pada video selama durasi yang telah ditentukan.

3.1.1 Gambaran Umum Metode Penelitian

Metodologi penelitian yang dilakukan pada penelitian ini dibagi menjadi tiga bagian yaitu: (1) pengumpulan data (2) deteksi tangan (3) pelacak tangan. Berikut diagram metodologi pada penelitian ini:



Pengumpulan data pada penelitian ini yakni mengumpulkan data berupa gambar dan video dimana data gambar merupakan gambar-gambar yang memiliki objek tangan di dalam gambar tersebut yang kemudian menjadi dataset yang akan digunakan pada tahap deteksi tangan dan data video merupakan gerakan objek yaitu tangan pada video dengan perpindahan posisi objek pada video tersebut yang akan digunakan pada tahap pelacak tangan pada penelitian ini. Pengumpulan data gambar akan dikerjakan selama skripsi 1 dan pengumpulan data video akan dikerjakan selama skripsi 2.

Setelah pengumpulan data, tahap berikutnya adalah tahap deteksi tangan yang akan dibagi menjadi empat bagian yaitu (1) pembuatan model (2) pelatihan model (3) validasi model (4) pengujian model. Pembuatan model deteksi tangan menggunakan dua buah arsitektur *convolutional neural network* yang nantinya akan dibandingkan antara kedua model yang memiliki akurasi yang lebih baik dalam mendeteksi tangan. Kemudian untuk melihat tingkat kesalahan pada masing-masing model, pengukuran tingkat kesalahan pada sebuah model dilakukan dengan cara *10-fold cross-validation*. Dan pada bagian pengujian model, jaringan yang telah dibangun diuji untuk mendeteksi tangan dengan data gambar tangan yang belum pernah di proses dan masuk ke dalam pelatihan model.

Setelah tahap deteksi tangan, tahap terakhir adalah pelacakan tangan. Pada tahap pelacakan tangan, tahap ini dibagi menjadi

3.2 Pengumpulan Data

Untuk mendukung penelitian ini, data merupakan variabel terpenting untuk menjadi bahan belajar model dalam mendeteksi dan melacak tangan. Pada model deteksi tangan, data yang diperlukan adalah gambar tangan dimana gambar tersebut harus memenuhi beberapa kondisi dimana gambar memiliki latar yang cukup kompleks dengan kata lain tidak polos, data merupakan gambar statis dan posisi tangan serta sikap tangan pada kumpulan data memiliki variasi yang cukup banyak. Selain gambar tangan diperlukan gambar bukan tangan untuk model belajar kebenaran. Gambar bukan tangan juga memiliki dimensi yang sama dengan gambar tangan namun untuk latar dan objek yang diambil dapat merupakan objek apapun kecuali tangan manusia, hal ini dilakukan untuk mendeteksi tangan dimana kondisi latar cukup kompleks. Pengumpulan data gambar untuk pelatihan model deteksi tangan akan dibagi menjadi dua bagian yaitu: (1) *training set* (2) *test set*. Untuk *test set*, data yang dikumpulkan akan memiliki perbedaan dimana data gambar akan memiliki latar polos dan memiliki satu kondisi dimana seluruh data memiliki tangan pada gambar. Sehingga nantinya pelatihan model akan menggunakan data yang memiliki latar yang kompleks dan memiliki kondisi gambar yang memiliki tangan dan gambar yang tidak memiliki tangan.

Model pelacakan tangan akan menggunakan data berupa gambar kontinu atau video sebagai masukan. Pengambilan data akan dilakukan dengan cara merekam gerak tangan dari tiga subjek berbeda kemudian data tersebut akan memiliki kondisi dimana data memiliki *frame per second* tidak lebih dari 30 fps. Masing-masing sampel dari video memiliki durasi tidak lebih dari satu menit dan

kondisi tangan dapat bebas bergerak dan berada di koordinat manapun dalam gambar. Postur dan bentuk tangan pada data video yang dikumpulkan harus sesuai dengan postur dan bentuk yang hanya ada pada dataset yang digunakan pada tahap deteksi tangan, hal ini dilakukan agar model deteksi mampu mendeteksi objek sesuai dengan bentuk dan postur tangan yang telah dipelajari model deteksi tangan. Rekaman yang diambil akan menggunakan kamera laptop dimana rekaman akan berisi gerakan tangan subjek yang bergerak bebas pada bidang spasial kamera laptop yang digunakan.

3.3 Deteksi Tangan

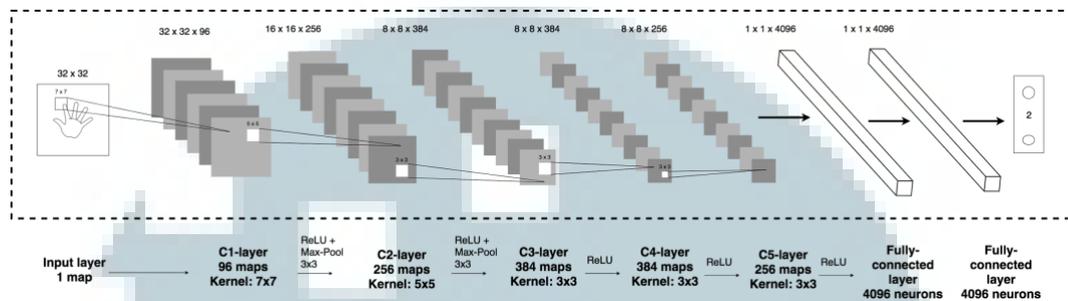
Terdapat empat bagian pada tahap deteksi tangan yaitu: (1) pembuatan model (2) pelatihan model (3) validasi model (4) pengujian model. Pada tahap ini, seluruh bagian dari tahap deteksi tangan akan menggunakan dua buah model yang berbeda yaitu model yang diadaptasi dari penelitian Chen, Wu, Hsieh & Fu (2016) dan model yang diadaptasi dari proyek *cuda-convnet2* Krizhevzky et al. (2012). Masing-masing bagian dari tahap deteksi tangan akan dilakukan menggunakan kedua model tersebut yang nantinya hasil dari kedua model tersebut dapat dibandingkan satu sama lain. Hasil deteksi menggunakan *convolutional neural network* dapat dikatakan baik jika tingkat generalisasi model cukup tinggi yaitu diatas 80%.

3.3.1 Pembuatan Model

Model untuk mendeteksi tangan pada penelitian ini menggunakan tensorflow sebagai library untuk dasar pembuatan model *deep learning*. Penelitian ini menggunakan dua buah model *deep learning* untuk mendeteksi tangan dan menggunakan metode *convolutional neural network* dengan arsitektur jaringan yang diadaptasi dari arsitektur yang dirancang oleh jaringan *convolutional neural network* dari penelitian Chen, Wu, Hsieh & Fu (2016) dan arsitektur jaringan *cuda-convnet2* untuk memproses data CIFAR-10 yang merupakan proyek yang dibangun oleh Alex Krizhevsky pada tahun 2011. Penelitian ini hanya mengikuti arsitektur dari kedua referensi, dimana penelitian ini hanya mengikuti jumlah *layer*, *neuron*, *pooling*, dan teknik normalisasi jika ada. Untuk pengaturan pada *hyperparameter* akan disesuaikan, dimana masing-masing model akan menggunakan *hyperparameter* berbeda sesuai dengan performa terbaik model terhadap *hyperparameter* yang digunakan.

Model pertama dibangun sesuai dengan arsitektur yang diadaptasi oleh penelitian Chen, Wu, Hsieh & Fu (2016), penggunaan model ini dikarenakan model deteksi tangan pada penelitian tersebut berhasil mendeteksi tangan dengan akurasi hampir 90%. Jaringan yang terdapat pada penelitian tersebut memiliki satu jaringan konvolusi dan tiga jaringan *neural network*. Tiga jaringan *neural network* tersebut antara lain: (1) *proposal network* (2) *detector network* (3) *estimator network*. Penelitian ini hanya menggunakan satu buah *neural network* diakhir jaringan konvolusi, jaringan *neural network* yang diimplementasikan adalah *detector*

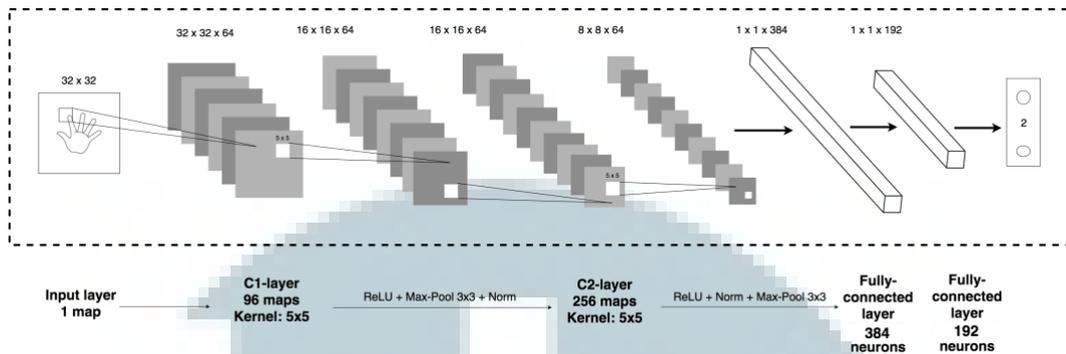
network. Pada jaringan deteksi tersebut berfungsi untuk mengenali *feature* hasil dari jaringan konvolusi apakah gambar tersebut merupakan tangan atau bukan tangan.



Gambar 3.1 Convolutional Neural Network Model Chen, Wu, Hsieh & Fu

(2016)

Jaringan yang diadaptasi dari penelitian Chent et al. 2016 (gambar 3.1) memiliki lima layer konvolusi dan dua layer *fully connected* yang kemudian pada ujung jaringan adalah hasil probabilitas sebuah gambar tangan atau gambar bukan tangan. Hasil tersebut diperoleh menggunakan fungsi *softmax* dengan distribusi probabilitas dari 0 sampai 1. Pada konvolusi pertama dan kedua, terdiri dari dua jendela konvolusi, dua fungsi non-linear ReLU, dan dua *pooling* layer *max-pooling*. Setiap *max-pooling* memiliki *kernel size* 3 x 3 piksel, *max-pooling* berfungsi untuk merangkum gambar masukkan dengan mengambil nilai maksimum dari kumpulan kotak pada jendela dan tidak saling tumpang tindih. Penggunaan fungsi non-linear ReLU digunakan untuk meningkatkan kecepatan pelatihan dan pembedaan fungsi *sigmoid* (Krizhevzky et al., 2012). Pada penelitian ini, setiap hasil konvolusi menggunakan *same-padding* atau memberi *padding* pada gambar masukkan dengan nilai konstan nol setiap piksel. *Zero-padding* digunakan untuk mendapatkan dimensi yang sama dengan gambar masukkan pada hasil konvolusi dalam jaringan.



Gambar 3.2 Convolutional Neural Network Model Cuda-Convnet2

Krizhevsky (2011)

Model kedua yang diadaptasi dari proyek *cuda-convnet2* untuk mengklasifikasikan dataset CIFAR-10 (gambar 3.2), penggunaan model kedua ini dikarenakan komputasi yang lebih efisien dilihat dari banyaknya layer pada model tersebut dan model ini mengadaptasi teknik normalisasi dari penelitian Krizhevsky et al. (2012) yang mampu meningkatkan generalisasi model. Jaringan tersebut memiliki dua buah layer konvolusi, dan dua buah layer *fully-connected*. Mirip dengan model jaringan pertama, pada model ini masing-masing konvolusi memiliki dua buah fungsi non-linear ReLU dan *max-pooling* 3x3. Namun, pada model ini, ada dua buah proses normalisasi diantara proses konvolusi. Normalisasi yang digunakan pada model tersebut adalah *local response normalization*. Normalisasi ini digunakan untuk normalisasi kecerahan, fungsi dari *local response normalization* sebagai alat bantu untuk generalisasi dan terbukti dapat mengurangi tingkat kesalahan saat melakukan *cross-validation* (Krizhevsky et al., 2012). Pada ujung jaringan pada model yang diadaptasi dari tensorflow memiliki fungsi yang sama dengan model pertama yaitu dengan mengimplementasikan fungsi *softmax* untuk menghasilkan dua buah kelas dengan tingkat distribusi probabilitas

berdasarkan label. Model kedua juga menerapkan *zero-padding* atau *same-padding* untuk mendapatkan dimensi yang sama dengan gambar masukan.

Kedua model yang akan digunakan pada penelitian ini memiliki kelebihan dan kekurangan masing-masing dari arsitektur model. Model pertama yang diadaptasi dari penelitian Chen, Wu, Hsieh & Fu (2016) memiliki arsitektur CNN yang cukup dalam yaitu memiliki lima buah layer, sehingga arsitektur tersebut dapat mempelajari semakin banyak corak pada setiap level terhadap objek pada gambar yang akan dideteksi. Namun, model pertama akan memakan komputasi yang lebih banyak dan karena memiliki banyak layer, model lebih mudah *overfitting* jika data memiliki kondisi dan keadaan objek tidak memiliki jumlah yang banyak. Model kedua yang diadaptasi dari proyek *cuda-convnet2* tidak memiliki layer yang sedalam pada model pertama, tetapi model kedua memiliki layer yang dapat menormalisasi hasil konvolusi dimana normalisasi dapat membantu meningkatkan performa model saat pelatihan tentunya dalam belajar mengenali corak. Model kedua juga tidak memakan komputasi yang banyak dibandingkan dengan model pertama. Namun, model kedua tidak dapat mengenali corak sebanyak model pertama sehingga akan sulit jika ingin mendeteksi keadaan atau kondisi objek yang beragam pada dataset.

Penggunaan dua buah model tersebut pada penelitian ini didasari untuk membandingkan performa kedua model dengan dalam mengklasifikasi dua buah kelas yaitu tangan dan bukan tangan. Model pertama yang memiliki layer yang dalam belum tentu lebih akurat dibandingkan dengan model kedua yang hanya memiliki dua layer saja. Semakin banyak *filter* yang belajar belum tentu

menghasilkan hasil dan performa yang paling baik. Hal utama yang dapat menyebabkan performa pada masing-masing model adalah data. Banyak faktor pada data yang dapat mempengaruhi performa mode seperti jumlah data, banyaknya jenis objek, banyaknya kondisi dan keadaan objek, dan lain sebagainya.

3.3.2 Pelatihan Model

Pada tahap ini, proses *data preprocessing* dan *data labelling* disisipkan bersama-sama dengan pelatihan model. Akan tetapi, proses tersebut dilakukan sebelum inialisasi variabel global. *Data preprocessing* menjadi langkah pertama dalam persiapan sebelum pelatihan model dan inialisasi variabel global, pertama pembacaan data pada masing-masing model menggunakan *script* python dan kemudian di proses menggunakan *library* open-CV untuk mengubah data gambar menjadi set matriks yang kemudian akan dikumpulkan kedalam satu array sesuai peran dataset. Seiring dengan pembacaan data dari *directory*, proses pemberian label untuk setiap data secara bersama dilakukan dengan proses pembacaan data. Proses *data labelling* yang dilakukan cukup sederhana karena masing-masing data sudah dikelompokkan berdasarkan kelas dan peran data tersebut baik sebagai data untuk pelatihan ataupun pengujian. Data untuk melakukan pelatihan model akan secara random diacak distribusinya untuk membantu generalisasi model.

Pelatihan model dilakukan dengan menggunakan *library* open-source *tensorflow* dan dimplementasikan pada bahasa pemrograman python. *Tensorflow* mendukung penggunaan GPU saat melakukan komputasi jaringan pada model

convolutional neural network. Pelatihan model dibagi menjadi dua bagian yaitu (1) pelatihan *convolutional neural network* Chen, Wu, Hsieh & Fu (2016) (2) pelatihan *convolutional neural network tensorflow CIFAR-10 based on cuda-convnet2*.

3.3.2.1 Pelatihan *Convolutional Neural Network* Chen, Wu, Hsieh & Fu (2016)

Pelatihan pada model ini merupakan *supervised training* yang menggunakan *stochastic gradient descent* dengan besar *minibatch* 32 dan mengimplementasi fungsi regularisasi *dropout* yang berfungsi untuk mengurangi *ovrefitting*. Jaringan pada model ini menggunakan *softmax cross-entropy* sebagai fungsi untuk menghitung jumlah *loss* dan sebagai aturan untuk memperbaharui parameter yang mampu belajar pada saat pelatihan.

Pelatihan model dibagi menjadi dua bagian yaitu pelatihan saat *cross-validation* dan pelatihan saat pengujian model. Pelatihan saat *cross validation* dan saat pengujian model dilakukan sebanyak 50 *epochs*, dimana satu *epochs* melakukan 112 iterasi untuk pelatihan saat *cross-validation* dan 120 iterasi untuk pelatihan saat pengujian model, pada setiap iterasi terdapat satu *minibatch* yang berisi satu kumpulan data dengan jumlah 32 gambar pada setiap kumpulan.

Model ini menggunakan nilai konstanta untuk *learning rate* adalah 0.000000003, dan probabilitas untuk melakukan *dropout* adalah 0.5. Selama pelatihan berlangsung, *learning rate* akan secara eksponensial menurunkan nilai konstanta pada λ 0.7 sebagai faktor penurun nilai *learning rate*. Penurunan nilai

konstanta pada *learning rate* dilakukan setiap 20 *epochs*. *Epoch num* berperan sebagai nilai eksponensial dimana nilai dari *epoch num* ditentukan dari jumlah *global step/decay step* (jumlah *batch* setiap *epoch* x 20). Peurunan *learning rate* dilakukan selama pelatihan model sampai seluruh *epochs* terselesaikan. Inisialisasi *weight* dilakukan dengan menggunakan distribusi normal dengan standard deviasi 0.005. Pelatihan model akan diikuti dengan kesimpulan nilai rata-rata akurasi dan *loss* di setiap iterasi dan *epochs*. Hal ini dilakukan untuk memudahkan penentuan kemampuan sebuah model setelah tahap pelatihan.

3.3.2.2 Pelatihan Convolutional Neural Network Tensroflow CIFAR-10 based on Cuda-Convnet2

Pelatihan model yang diadaptasi dari tensorflow sama dengan model pertama dimana pelatihan model ini merupakan *supervised training* dan menggunakan *stochastic gradient descent* dengan besar minibatch 32 serta menggunakan fungsi standard regularisasi *L2-regularization error*. Jaringan pada model ini menggunakan fungsi *softmax cross-entropy* yang berperan untuk menghitung jumlah *loss* saat pelatihan dan sebagai aturan untuk memperbaharui paramter dalam model yang mampu belajar atau dapat beradaptasi. Pelatihan *cross validation* dan saat pengujian model dilakukan sebanyak 40 *epochs*, dimana satu *epochs* melakukan 112 iterasi untuk pelatihan saat *cross-validation* dan 120 iterasi untuk pelatihan saat pengujian model, pada setiap iterasi terdapat satu minibatch yang berisi satu kumpulan data dengan jumlah 32 gambar pada setiap kumpulan.

Model ini menggunakan nilai konstanta untuk *learning rate* sebagai *hyperparameter* adalah 0.00001 sebagai nilai awal dalam pelatihan model. *Learning rate* dalam pelatihan model ini juga akan secara eksponensial menurunkan nilai konstanta dari nilai awal *learning rate*. Prosedur penurunan *learning rate* sama dengan model sebelumnya yang diadaptasi dari penelitian Chent et al. (2013). Nilai λ sebagai faktor penurun nilai *learning rate* dalam pelatihan model ini adalah 0.96. Penurunan nilai *learning rate* dilakukan untuk mencegah *vanishing gradient* dimana model gagal untuk mencapai *local minimum* saat pelatihan. Penurunan nilai *learning rate* akan dilakukan setelah 12 *epochs* terselesaikan. Fungsi standard regularisasi L2-*regularization error* pada model ini memiliki nilai β 0.04, fungsi tersebut digunakan pada seluruh nilai *weight* di layer pertama dan kedua *fully-connected*. Inisialisasi *weight* dilakukan dengan menggunakan distribusi normal dengan standard deviasi 0.005. Model ini juga menggunakan teknik normalisasi pada proses konvolusi yaitu *local response normalization* untuk memperjelas gambar saat diproses oleh komputer.

3.3.3 Validasi Model

Tahap validasi model adalah tahap untuk menseleksi model yang telah dilatih menggunakan dataset yang telah dikumpulkan. Validasi model dilakukan menggunakan kedua model yang digunakan dalam penelitian ini dan di validasi menggunakan 10-*fold cross validaiton*. Kedua model akan divalidasi menggunakan 10-*fold* yang kemudian hasil dari 10-*fold* masing-masing model akan dibandingkan

satu sama lainnya. Untuk melakukan *10-fold cross validation* pada penelitian ini, pembahasan lengkap akan dibahas pada subbab 3.3.3.1.

3.3.3.1 10-fold Cross-Validation Model

Cross-validation dilakukan pada penelitian ini untuk melakukan seleksi model dan menyimpulkan performa dari sebuah model. Penelitian ini menggunakan *10-fold cross-validation* sebagai metode validasi, dua model yang telah dibangun akan divalidasi menggunakan *10-fold* dari jumlah data dari dataset. Proses validasi ini akan dilakukan bersamaan dengan pelatihan model sehingga pelatihan model akan memiliki 10 kali iterasi.

Dataset pelatihan akan dipecah terlebih dahulu menjadi 10 bagian, banyaknya data untuk pelatihan model akan dibagi berdasarkan rumus *k-fold validation* yaitu $n = N - N/k$ dan banyaknya data untuk melakukan pelatihan adalah n , karena $k = 10$ maka hasilnya 10 set data yang akan digunakan untuk melakukan *cross-validation*. Dan sebanyak N/k akan menjadi data untuk melakukan validasi. Proses ini dilakukan sebelum inisialisasi global variabel saat pelatihan model CNN. Pembagian *fold* akan dilakukan menggunakan dataset pada *training-set* dimana setelah dibagi masing-masing *fold* akan memiliki dua set data yaitu (1) *training set* (2) *validation set*

Tahap ini dibagi menjadi dua bagian yaitu: (1) *10-fold cross-validation CNN model* Chen, Wu, Hsieh & Fu (2016) (2) *10-fold cross-validation CNN model*

tensorflow based on cuda-convnet2. Kedua model akan divalidasi menggunakan 10-fold dengan distribusi data yang sama. Hasil dari *cross-validation* ini adalah *error rate* model dari 10 validasi yang dilakukan setelah 10 kali pelatihan model dan akurasi model dalam mengklasifikasikan gambar. Hasil *cross-validation* untuk kedua model akan digunakan sebagai laporan di bab berikutnya dalam penelitian ini.

3.3.4 Pengujian Model

Pada tahap ini kedua model akan diuji akurasinya dengan menggunakan dataset *test set*. Pengujian ini dilakukan untuk melihat kemampuan generalisasi pada masing-masing model pada saat *cross-validation* dan tanpa *cross-validation*. Pada tahap ini, model akan diuji menggunakan dua skenario yaitu yang pertama dengan menguji model menggunakan dataset *test set* setelah *cross validation* pada setiap *fold*. Yang kedua, model akan diuji penuh dengan seluruh data pada *training set* untuk pelatihan model dan diuji menggunakan data pada dataset *test-set*. Pembahasan lengkap dari dua skenario tersebut akan dibahas di subbab 3.3.4.1.

3.3.4.1 Pengujian Akurasi Model

Pada skenario pertama, yaitu model yang telah dilatih dan divalidasi menggunakan data pada sebuah *fold* yang telah dibagi, model tersebut akan diuji menggunakan dataset dari *test set* yang merupakan data terpisah dari data pada *fold*

yang dibagi. Sehingga, pengujian juga akan terjadi sebanyak 10 kali. Skenario kedua yaitu model diuji penuh dimana model tidak menggunakan data pada sebuah *fold* melainkan menggunakan seluruh data pada kecuali data pada *test set* untuk digunakan dalam melatih model. Setelah model dilatih dengan seluruh data kecuali data pada *test set*, data akan diuji akurasi menggunakan data pada dataset *test set* dimana seluruh data merupakan gambar yang memiliki objek tangan. Pengujian pada skenario kedua akan dilakukan sebanyak tiga kali. Pengujian menggunakan dua skenario berbeda ini dikarenakan untuk melihat kemampuan model setelah validasi apakah model juga dapat melakukan klasifikasi dengan performa yang sama saat validasi dan juga untuk melihat perbedaan model dalam mengklasifikasi data pada dataset *test set* yang memiliki kondisi gambar berbeda dimana model dilatih dengan seluruh data kecuali data pada *test set* dan model juga dilatih dengan sebagian data yang merupakan salah satu *fold*. Sehingga pelatihan pada skenario dua akan menggunakan data yang lebih banyak dibandingkan data pelatihan untuk skenario pertama. Oleh karena itu, dengan perbedaan jumlah data pada pelatihan hasil dari pengujian akurasi pada masing-masing skenario akan memiliki performa yang berbeda dalam mengklasifikasi data pada dataset *test set*.

3.4 Pelacak Tangan

Terdapat dua bagian pada tahap pelacak tangan yaitu: (1) pembuatan model (2) pengujian akurasi model pelacak tangan. Pada bagian pertama, model pelacak tangan akan dibuat menggunakan model deteksi tangan yang telah dilatih

sebelumnya yang digunakan untuk melakukan lokalisasi tangan pada sebuah video. Model pelacak tangan juga menggunakan *image pyramid* dan *sliding window* untuk mendukung model deteksi tangan yang telah dilatih dalam melakukan lokalisasi tangan pada video. Untuk menguji akurasi dari model pelacak tangan yang dibuat, akan dihitung menggunakan *euclidean distance* antara *centroid* dan titik koordinat lokalisasi.

3.4.1 Pembuatan Model

Model pelacak akan menggunakan *gaussian pyramid* sebagai alat untuk membantu model deteksi mengetahui keberadaan dan posisi tangan pada gambar. Perlunya *gaussian pyramid* dikarenakan dimensi gambar pada video yang tidak sama dengan dimensi *input* pada model deteksi tangan yang telah dilatih dan kemungkinan posisi serta dimensi dari tangan pada video yang juga tidak sama dengan dimensi *input* model deteksi. Model deteksi yang digunakan untuk model pelacak tangan akan dilatih kembali menggunakan data yang telah dikumpulkan sebelumnya, dan data gambar tangan yang lebih banyak dan lebih variatif dari segi postur, posisi, dan resolusi. Pada proses pelatihan model deteksi yang dilakukan, ada improvisasi dalam penggunaan dataset yang telah dikumpulkan sebelumnya diluar *test set*. Seluruh data tangan akan di augmentasi atau di rotasi 90° , 180° , dan 270° .

Gaussian pyramid pada model pelacak akan menggunakan nilai $\sigma = 1$ untuk menghaluskan gambar sebelum gambar di *downsmample* sebanyak setengah

kali dari gambar dan menggunakan *gaussian filter*. Secara umum penggunaan *gaussian pyramid* akan melakukan *downsample* pada gambar sebanyak 5 kali sehingga akan ada 5 buah gambar yang akan menjadi objek untuk di deteksi menggunakan model deteksi tangan yang telah dibuat sebelumnya. Kecuali, jika dimensi *sample* pada gambar yang telah di *downsample* lebih kecil dari dimensi *input* model deteksi tangan maka proses *downsample* akan berhenti pada *sample* sebelumnya.

Selain *image pyramid*, model pelacak tangan akan menggunakan *sliding window* dengan *window size* berukuran 25% lebar dan 25% tinggi dari *i_{th} image pyramid* (*minimum size 25% of image*) dan *step size 50 pixel*, jika *i_{th}* memiliki lebar dan tinggi lebih kecil dari *minimum window size * 3*, maka besarnya *window size* adalah 32 x 32 sesuai dengan ukuran *input window* pada model deteksi tangan yang telah dilatih. Setiap *window size* yang tidak memiliki ukuran sesuai dengan *input window* pada model deteksi tangan pada *sliding window* di *i_{th} image pyramid*, maka hasil *window* akan diubah ukurannya menjadi 32 x 32 agar dapat dideteksi menggunakan model deteksi tangan yang telah dilatih sebelumnya.

Model pelacak tangan akan menampung seluruh kemungkinan dari seluruh *sliding window* untuk seluruh *pyramid* dengan nilai probabilitas sebagai tangan diatas sama dengan 85% yang nantinya akan digunakan untuk lokalisasi. Hasil yang ditampung merupakan *average pixel value*, *true coordinates* dan *true shape* sesuai dengan ukuran asli dari gambar dimana *x*, *y*, *w*, *h* pada *window* yang berhasil dideteksi dan memiliki probabilitas sebagai tangan diatas sama dengan 85% akan di *upsample* sampai 2x lipat dari *i_{th} pyramid* sampai ke ukuran asli pada gambar

sehingga menghasilkan *true window* pada ukuran gambar asli. *True window* tersebut yang akan menjadi parameter untuk membuat *bounding box* pada tangan yang telah dilokalisasi.

Untuk menangani kesalahan deteksi yang dilakukan oleh model deteksi tangan dan telah ditampung sebelumnya, seluruh hasil deteksi yang telah ditampung akan di segmentasikan. Hal ini dilakukan untuk mencari densitas *feature* dan lokasi *window* yang paling sering terdeteksi. Segmentasi akan menggunakan 3 algoritma *clustering* yaitu *k-mean*, *dbscan*, dan *mean shift*. Penggunaan 3 algoritma ini digunakan untuk membandingkan masing-masing algoritma untuk mendukung model pelacak tangan yang dibuat mencapai performa yang terbaik. Setiap algoritma akan diuji menggunakan 5 parameter yang berbeda. Jika pada suatu algoritma memiliki parameter lebih dari 1 maka pengujian masing-masing algoritma akan dilakukan sebanyak $n * 5$ dimana n merupakan jumlah parameter dan sistem perubahan parameter adalah *static* dimana parameter ke i_{th} yang akan berubah-ubah nilai dan sisanya akan ditentukan berdasarkan *default implementation* dari *library* yang digunakan.

Penentuan nilai parameter yang akan digunakan untuk menguji algoritma *clustering* yang digunakan akan ditentukan *incremental*, untuk *k-means* banyaknya k yaitu mulai dari 2-6, untuk *dbscan* besarnya *epsilon* yaitu 0.2-0.6 dan besarnya sample minimal yaitu 2-6, dan untuk *meanshift* nilai *quantile* yaitu dari 0.2-0.6. Alasan dari banyaknya k pada *k-means* dimulai dari 2-6 adalah dimana jumlah *data points* hanya 3 dan jika k terlalu besar akan menyebabkan hasil *clustering* yang sangat menyebar maka dari itu akan dilakukan uji coba terhadap parameter ini.

Besarnya *epsilon* yang memiliki nilai dibawah 0 dikarenakan hasil *standard score* yang memiliki nilai antara 0-1 sehingga radius *core points* dengan *neighbor* pada *dbscan* juga harus dibawah 0. Pada algoritma *meanshift*, *quantile* merupakan seberapa banyak pergeseran yang dilakukan terhadap *mean* pada setiap *data points*. Pengaturan parameter ini harus diantara 0-1 dimana 0-1 adalah probabilitas. Jika nilai nya 0.5 maka pergeseran yang dilakukan suatu *data point* adalah 50% dari *mean* yang didapat dari seluruh *data points* dalam sebuah radius. Penentuan radius dan jumlah *data points* untuk algoritma *mean shift* akan menggunakan *default implementation* dari *library scikit-learn*. Hal ini dikarenakan *default implementation* dari *library* tersebut sudah baik, teruji, dan sudah digunakan secara massal.

Data points yang akan digunakan adalah *mean* dari hasil tangkapan *window*, titik *x*, titik *y*, lebar dan tinggi. Kemudian seluruh variabel akan di normalisasi menggunakan *standard score*. Hasil dari segmentasi tersebut akan digunakan untuk mencari *cluster* mana yang memiliki jumlah paling banyak, kemudian dari seluruh entitas pada *cluster*, *data points x, y, w, h* akan dicari rata-rata nya dan hasil tersebut akan digunakan sebagai posisi yang akan digunakan untuk lokalisasi. Dasar dari penggunaan metode ini dikarenakan untuk mengantisipasi hasil deteksi yang salah dan jika posisi dari hasil metode ini menghasilkan lokalisasi yang salah yaitu *bounding box* tidak pada tangan pada video, maka model deteksi gagal untuk mendeteksi tangan pada *sample* video dan pelacakan tangan tidak dapat terjadi.

Setelah mendapatkan posisi untuk lokalisasi dan *bounding box*, model pelacak tangan akan menggunakan algoritma *mean shift* untuk melacak tangan

sepanjang durasi video dimana area daripada *bounding box* merupakan *region of interest*. Sebelum menggunakan algoritma *mean shift* setiap *frame* akan diubah ke format HSV dikarenakan perbedaan warna pada format HSV lebih mencolok dan menggumpal sehingga memudahkan pelacakan objek yang berada pada *region of interest*.

3.4.2 Pengujian Akurasi Pelacakan Tangan

Pengujian akurasi model pelacak tangan dilakukan dengan cara menghitung jarak lokalisasi yang telah dilakukan dengan titik tengah (*centroid*) yang didapat dari rata-rata dari seluruh kumpulan koordinat dari *window* yang berhasil terdeteksi pada *frame* yang digunakan untuk melakukan inisial lokalisasi dalam setiap sample yang berupa video. Perhitungan jarak tersebut akan dihitung menggunakan *euclidean distance*. Semakin kecil jarak antara lokalisasi dengan *centroid* maka model dapat dikatakan lebih akurat dibandingkan dengan hasil jarak yang lebih besar. Model deteksi tangan akan di validasi kembali menggunakan *10-fold cross validation* dan diuji kembali dengan dataset NUS I yang merupakan *test set* pada pelatihan model deteksi sebelumnya. Hal ini dikarenakan jumlah data dan data yang diberikan untuk melatih model deteksi tangan berbeda dari sebelumnya.