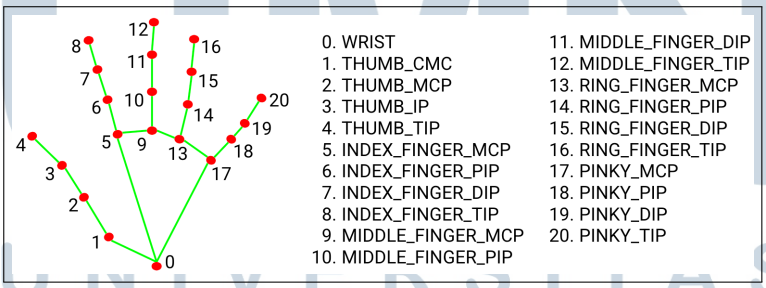
BAB 2
LANDASAN TEORI

2.1 MediaPipe

MediaPipe adalah sebuah *framework* untuk membangun *pipeline* yang dapat melakukan inferensi atas data sensorik arbitrer. Dengan MediaPipe, *pipeline* dapat dibangun sebagai *graph* komponen modular, model inferensi, algoritma pemrosesan media maupun transformasi data. Data sensorik seperti data audio atau video yang masuk ke dalam *graph*, akan diproses oleh MediaPipe, misalnya dengan *object-localization* sehingga dapat menghasilkan *output* berupa *hand-landmark* dari *graph* [20]. MediaPipe menyediakan banyak solusi *open source* yang dapat digunakan, salah satunya adalah MediaPipe Hands yang dapat digunakan untuk *hand tracking*.

2.2 MediaPipe Hands

MediaPipe Hands menggunakan beberapa model agar dapat menghasilkan sebuah *landmark*. *Palm detection model* yang beroperasi pada gambar dan mengembalikan *output* berupa kotak yang menjadi batas orientasi suatu tangan. *Hand landmark model* yang beroperasi di dalam wilayah kotak dari gambar yang sudah didefinisi oleh *palm detector*, dan mengembalikan titik koordinat 3D dengan fidelitas tinggi [21].



Gambar 2.1. Gambar *landmark* tangan dari MediaPipe Hands

Untuk mendapatkan data *ground truth*, MediaPipe Hands menggunakan data sebanyak ∼30.000 gambar yang disertai dengan 21 3D koordinatnya, 21 titik tersebut dapat dilihat pada Gambar 2.1. Koordinat Z didapatkan dengan memetakan *image depth* yang dimiliki oleh masing-masing koordinat. MediaPipe Hands dapat menyajikan *output* dalam bentuk 3D walau hanya menerima *input 2D* dengan

cara mengestimasi *relative depth* dari tangan yang terdeteksi [1]. Solusi *Hand Tracking* dari MediaPipe juga memiliki *calculators* yang dapat menjalankan suatu komputasi *neural network* dengan lebih *optimized* sehingga dapat menggunakan GPU *acceleration* pada suatu *mobile device* lebih efektif [1].

2.2.1 Tensorflow dan Keras

Tensorflow adalah *framework open source* untuk deep learning yang dikembangkan oleh Google. Tensorflow juga merupakan sebuah *symbolic math library* yang digunakan untuk *neural network* dan paling cocok digunakan untuk *dataflow programming* di berbagai bidang. Tensorflow juga menyediakan *multiple abstraction level* untuk membangun dan melatih sebuah model [22]. Tensorflow juga dapat mengkonversi suatu model menjadi tflite dengan cara *Quantization*, mengubah value *base 64 float* pada model, menjadi value integer. Hasil training model yang telah dikonversi menjadi tflite seharusnya sudah dapat digunakan untuk melakukan inferensi pada *device smartphone*, karena model tflite memang ditujukan untuk melakukan inferensi pada suatu *mobile device* [23].

Keras adalah sebuah *high-level neural network* API yang ditulis dengan Python. *library open source* ini didesain agar dapat menyediakan environment untuk melakukan eksperimen terhadap suatu *deep neural network* dengan cepat, dan dapat berjalan diatas Microsoft Cognitive Toolkit (CNTK), TensorFlow dan Theano [22].

Untuk membangun model klasifikasi *hand gesture*, digunakan Keras *sequential model* sebagai struktur model. *Sequential model* menumpuk *layer*, dimana tiap *layer* memiliki satu *input tensor* dan satu *output tensor* [24].

2.2.2 Dense Neural Network

Dalam Keras, Dense Neural Network ini disebut juga dengan Multi Layer Perceptron (MLP) [25]. Arsitektur dari Dense Neural Network ini dibentuk dengan menggunakan komponen *layer* berikut yang ada di Tensorflow Keras

1. Dropout
   Dropout *layer* menentukan *input units* menjadi 0 dengan frekuensi *rate* dalam tiap tahap saat *training* data, ini dapat mencegah terjadinya *overfitting* [26][27].

2. Dense

   Dense *layer* menjalankan operasi dibawah ini dengan menerima sebuah *input* dan *output* yang didapatkan dan dipengaruhi oleh jumlah *neuron / units* yang ditentukan di Dense *layer*. Misalnya jika *shape* dari *input* berupa (None, 16) maka *shape* dari *output* menjadi (None, 32) [28].

$$output = activation(dot(input, kernel) + bias) \tag{2.1}$$

Kemudian untuk *activation*, digunakan dua jenis *activation* berikut:

1. Rectified Linear Units (ReLU)

   RELU digunakan untuk membantu model dalam menentukan *interactions effects* dan menyelesaikan masalah *non-linear*. *Interactions effects* adalah keadaan dimana suatu variabel A mempengaruhi hasil prediksi dengan cara yang berbeda tergantung terhadap variabel B. Masalah *non-linear* adalah keadaan dimana suatu *slope* tidak berbentuk garis lurus atau dalam satu aksis saja [29].

$$f(x) = max(0, x) \tag{2.2}$$

2. Softmax

   Softmax adalah fungsi yang mengkonversi vektor angka menjadi vektor probabilitas, dimana probabilitas tiap nilai proporsional dengan skala relatif dari tiap nilai vektor. Dalam machine learning, Softmax digunakan sebagai *activation function* dalam suatu model *neural network*. Khususnya dalam masalah *multinomial* untuk memberikan *output* nilai N. Fungsi Softmax menormalisasi *output*, mengkonversi nilainya dari *weighted sum* menjadi probabilitas. Nilai dari tiap *output* dari fungsi *softmax* diinterpretasikan sebagai probabilitas dari sebuah *class* [30].

$$softmax(z_i) = \frac{e^z_j}{\sum e^e_j} \tag{2.3}$$

Terakhir menggunakan sebuah fungsi *loss*, Categorical Cross-Entropy Loss. Categorical Cross-Entropy Loss melatih Dense Neural Network untuk menghasilkan *output* probabilitas dari sebuah *C* class. Ini juga digunakan untuk klasifikasi *multi-class*. Kasus yang umum dari klasifikasi *multi-class* biasanya

memiliki label dengan representasi *one-hot*, jadi hanya *class* positif C_p yang tersimpan di dalam *loss* [31].

2.2.3 User Datagram Protocol

User Datagram Protocol (UDP) adalah protocol yang digunakan dalam jaringan internet untuk mengirimkan data yang sensitif terhadap waktu, seperti misalnya *video* atau DNS *lookup*. UDP tidak perlu membuat koneksi secara utuh *(three way handshake)* sebelum *data* dapat ditransfer. Ini membuat data dapat ditransfer dengan cepat, tetapi juga beresiko membuat data hilang dalam prosesnya dan rentan terhadap serangan Distributed Denial-of-Service (DDoS) [32]. Penelitian kali ini menggunakan UDP untuk mengirimkan data secara *local*, dari Python ke Unity. Jadi tidak perlu khawatir dengan adanya serangan siber.

2.2.4 JavaScript Object Notation

JavaScript Object Notation (JSON) adalah format file / penukaran data yang menggunakan *text* yang dapat dibaca dengan mudah oleh manusia. JSON ini dapat digunakan dalam pertukaran data elektronik yang luas karena didukung oleh hampir semua bahasa pemrograman [33].

2.2.5 Metrik Evaluasi

Model yang dibuat memiliki *multiple label* atau *class*, beberapa metrik evaluasi yang dapat digunakan untuk mengevaluasi performa dari model klasifikasi dengan *multiple label / classes* adalah *accuracy, precision, recall* dan *F1-Score* [34]

1. *Accuracy*

   Accuracy langsung dihitung dari hasil *confusion matrix*. Untuk menghitung *accuracy*, dipertimbangkan jumlah elemen True Positive (TP) dan True Negative (TN) pada pembilang dan jumlah semua *confusion matrix* pada penyebut. TP dan TN adalah elemen yang diklasifikasikan oleh model dan yang berada di diagonal utama pada *confusion matrix*, penyebut juga mempertimbangkan semua elemen di luar diagonal utama yang salah diklasifikasikan oleh model, False Positive (FP) dan False Negative (FN).

Rumus dari *accuracy* adalah:

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

2. *Precision*

   Precision adalah pembagian elemen True Positive dibagi dengan jumlah kolom total yang diprediksi positif. True Positive adalah elemen yang diberi label positif oleh model dan memang positif, sedangkan False Positive adalah elemen yang diberi label positif oleh model, tetapi sebenarnya negatif. *Precision* memberitahu seberapa besar model dapat dipercaya ketika model memprediksi *input* sebagai positif. Rumus dari *precision* adalah:

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

3. *Recall*

   Recall adalah pembagian elemen True Positive dibagi dengan jumlah baris total dari yang diprediksi positif. False Negative adalah elemen yang telah diberi label negatif tetapi sebenarnya positif oleh model. *Recall* mengukur akurasi model dalam memprediksi *class* yang positif, secara intuitif mengukur kemampuan model dalam menemukan semua unit positif dalam dataset. Rumus *Recall* adalah:

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

4. *F1-Score*

   F1-Score mulai menilai kinerja model dalam melakukan klasifikasi dari melihat *confusion matrix*, dengan menggabungkan *Precision* dan *Recall* dalam konsep *harmonic mean*.

$$F1 - Score = \left(\frac{2}{precision^{-1} + recall^{-1}}\right) = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall}\right) \tag{2.7}$$

   Rumus *F1-Socre* dapat diartikan sebagai *weighted average* antara *Precision* dan *Recall*, dimana *F1-Score* mencapai nilai terbaiknya pada 1 dan terburuknya pada 0. Kontribusi dari *precision* dan *recall* ke dalam *F1-Score*

relatif sama dan *harmonic mean* berguna untuk mendapatkan titik *trade-off* terbaik dari keduanya.