BAB II
LANDASAN TEORI

2.1 Sistem Isyarat Bahasa Indonesia

Menurut Tonny (Zen, 2017), satu kosa isyarat dalam Sistem Isyarat Bahasa Indonesia (SIBI) adalah kata yang bisa diberikan imbuhan untuk memberikan sebuah kata dasar banyak arti. Sebagai contoh, kosa kata ajar bisa diberi imbuhan agar menjadi "belajar" atau "diajar".

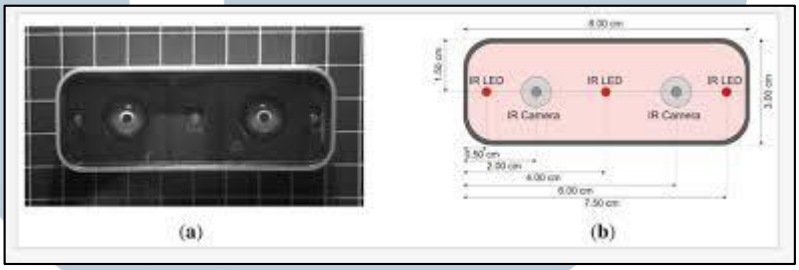
Departemen Sosial (2010) menjelaskan bahwa Sistem Isyarat Bahasa Indonesia memiliki 26 abjad jari dan 10 isyarat angka. Terdapat tiga penggunaan abjad jari menurut Departemen Sosial (2010) yang dijelaskan sebagai berikut.

1. Digunakan untuk mengisyaratkan nama orang.
2. Digunakan untuk mengisyaratkan nama-nama daerah/kota.
3. Digunakan untuk mengisyaratkan nama-nama benda yang belum mempunyai isyarat.

Gambar 2.1 Tabel abjad jari Sistem Isyarat Bahasa Indonesia (Depsos, 2010)

2.2 Leapmotion

Pada penelitian ini, *Leapmotion* digunakan sebagai perangkat untuk merekam citra digital bahasa isyarat tangan. Leapmotion didukung oleh 3 pemancar sinar inframerah dan 2 kamera penerima sinyal inframerah. Hasil citra yang dihasilkan merupakan gambar *grayscale* diikuti dengan pantulan gelombang cahaya pada spektrum yang sama (Pavaloiu, 2016). Arsitektur dari perangkat *Leapmotion* dapat dilihat pada gambar berikut.



Gambar 2.2 Struktur kamera *Leapmotion* (Shao, 2017)

2.3 Gaussian Blur

Gaussian blur adalah sebuah filter *low-pass* yang sering digunakan pada tahap *preprocessing* citra digital sebelum diolah oleh jaringan saraf tiruan. Bobot yang digunakan pada kernel konvolusi *gaussian* tidak sama, berkurang dari kernel paling tengah hingga ujung (Saleh dkk., 2010). Dodge dan Karam (2016) melakukan penelitian mengenai dampak dari lima jenis kompresi gambar dan *noise* terhadap hasil akurasi dari *neural network*. Pada penelitian tersebut, empat *neural network* yang diuji dengan dataset ImageNet menunjukan sensitivitas yang lebih tinggi terhadap *blur* dan *noise*, sementara disrupsi dari kontras dan kompresi gambar tidak terlalu berpengaruh banyak. Dalam penelitian ini, *Gaussian Blur* akan

diimplementasikan pada tahap *image preprocessing* sebelum digunakkan sebagai bahan training pada algoritma Convolutional Neural Network.

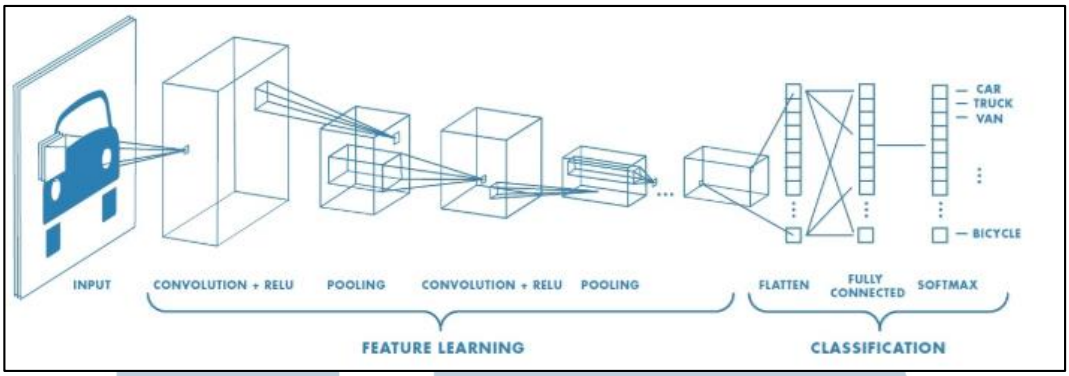
Kernel konvolusi yang akan digunakan dalam penelitian ini memliki ukuran 5x5, yang akan digunakan untuk menghitung nilai konvolusi. Setelah nilai konvolusi dihitung, angka tersebut dibagi dengan 256 untuk mendapatkan nilai pixel yang baru. Kernel yang digunakan dalam penelitian ini dapat dilihat pada Gambar 2.4.

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gambar 2.4 Kernel konvolusi 5x5

2.4 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah salah satu jenis dari jaringan saraf tiruan yang biasa digunakan pada citra digital (Sena, 2017). CNN dirancang untuk memudahkan pengenalan pola pada citra digital. Hasil *encode* dari sebuah fitur citra digital dapat dimasukkan kedalam arsitektur CNN, sehingga jaringan menjadi lebih cocok untuk deteksi objek pada citra digital serta mengurangi jumlah parameter yang diperlukan (O'Shea dan Nash, 2015). Arsitektur CNN secara singkat dapat dilihat pada Gambar 2.3.



Gambar 2.3 Arsitektur jaringan syaraf tiruan CNN (Sena, 2017)

Wu (2017) menjelaskan 4 operasi utama dalam CNN sebagai berikut.

1. *Convolution*

Tujuan utama pada tahap ini adalah untuk mengekstraksi fitur yang ada pada citra digital. Setiap gambar dikonversi menjadi representasi data dalam bentuk matriks dan melakukan perkalian matriks untuk tiap kombinasi yang ada. Hasil dari perhitungan tersebut merupakan hasil konvolusi dari fitur dominan pada tiap gambar. Nilai pengali dari index kernel dapat diubah untuk mendapatkan hasil ekstraksi fitur yang berbeda.

2. *Non-Linearity* (ReLu)

Rectified Linear Unit (ReLu) memastikan bahwa tidak ada index yang bernilai negatif setelah tahap *convolution* dijalankan. Semua index negatif diganti nilai nol. Hasil dari tahap ini disebut sebagai *Rectified Feature Map*.

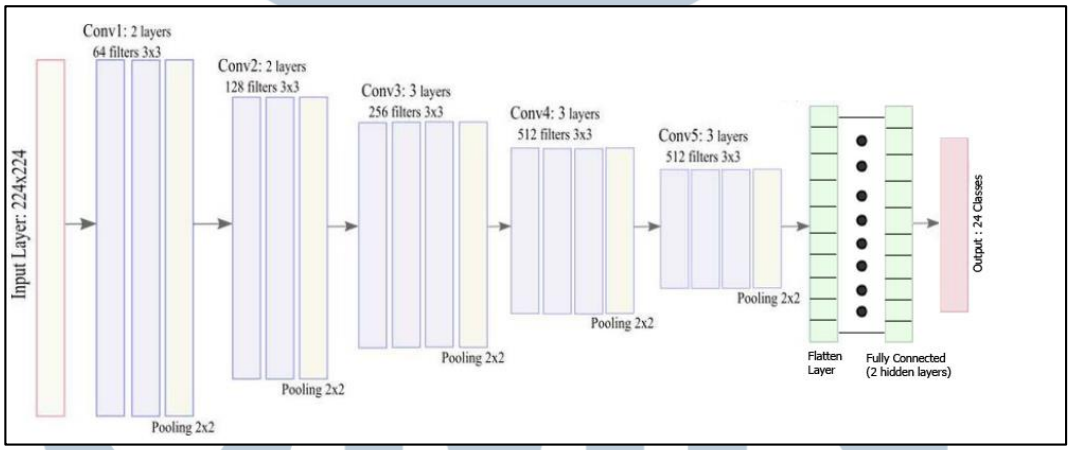
3. *Pooling*

Tahap *pooling* dilakukan untuk mengambil informasi paling penting dari fitur yang sudah di ekstraksi. Metode yang sering digunakan pada tahap ini adalah *max pooling* dimana nilai tertinggi dari matriks yang dipilih menjadi representasi titik pada matriks akhir. Tahap *convolution, non linearity* dan *pooling* dapat diulang hingga fitur yang diinginkan dapat terlihat.

4. *Classification*

Tahap *classification* menentukan kelompok dari sebuah matriks setelah semua fiturnya berhasil di ekstraksi. Sistem akan membuat sebuah prediksi berdasarkan probabilitas kemiripan dengan gambar yang sudah dikelompokkan sebelumnya.

Pada penelitian ini, algoritma Convolutional Neural Network dibangun berdasarkan arsitektur VGG16. Visual Geometry Group 16 (VGG16) dirancang oleh Visual Geometry Group dari Universitas Oxford, 16 pada nama VGG16 melambangkan jumlah layer yang dimiliki yaitu sebanyak 16 layer. VGG16 memiliki 1 neuron masukan, 5 blok konvolusi yang terdiri dari layer convolution, ReLu dan Pooling, dan 3 dense layer (Simonan dan Zisserman, 2014). Arsitektur VGG16 dapa dilihat pada gambar 2.4.



Gambar 2.4 Arsitektur VGG16

2.5 F-score Evaluation

Performa dari sebuah sistem dapat diukur dengan melihat nilai presisinya, biasanya diindikasikan menggunakan sebuah angka (Gaussier dan Goutte, 2014). F-score merupakan salah satu tolak ukur untuk menilai tingkat akurasi sebuah

classifier, yang menunjukan seberapa tepat hasil klasifikasi yang dilakukan (berapa kasus yang diprediksi dengan benar) serta seberapa kuat hasil yang diprediksi (tidak banyak kasus yang salah diprediksi) (Mishra, 2018). Untuk dapat menghitung F1-score, *precision* dan *recall* harus dihitung terlebih dahulu.

2.5.1 Precision

Precision adalah jumlah dari hasil prediksi positif dibagi dengan jumlah prediksi positif dan negatif yang berhasil diprediksi oleh *classifier* yang dibangun (Mishra, 2018). Berikut adalah rumus untuk menghitung *precision*.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad ...(2.1)$$

2.5.2 Recall

Recall adalah jumlah dari hasil prediksi positif dibagi dengan jumlah dari semua sampel yang relevan (semua sampel yang diidentifikasi sebagai positif) (Mishra, 2018). Berikut adalah rumus untuk menghitung *recall* .

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad ...(2.2)$$

2.5.3 F1-Score

F1-Score dapat dihitung setelah mendapatkan nilai dari *precision* dan *recall*. Nilai *precision* yang tinggi dan *recall* yang rendah akan menghasilkan akurasi yang tinggi, tetapi sistem akan cenderung kesulitan ketika mengklasifikasi kasus yang sulit diidentifikasi. Semakin besar nilai *F1-Score* semakin bagus performa dari model yang dirancang (Mishra, 2019). Berikut adalah rumus untuk menghitung *F1-Score*.

$$F1 - score = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \quad ...(2.3)$$