BAB 2
LANDASAN TEORI

2.1 Bahasa Isyarat Indonesia (BISINDO)

Bahasa Isyarat Indonesia (BISINDO) merupakan medium komunikasi resmi yang digunakan oleh tuna runggu dan tuna wicara di Indonesia. Bahasa ini mengandalkan kombinasi pergerakan tangan, raut muka, serta pergerakan pose tubuh. Bahasa ini mengadopsi dari bahasa isyarat amerika [19] yang mengalami campuran lokal dari beberapa kota. Bahasa ini merupakan perkembangan dari Sistem Isyarat Bahasa Indonesia (SIBI) yang lebih mencerminkan kebutuhan dan budaya lokal [20]. Tidak seperti SIBI yang menggunakan satu tangan, BISINDO menggunakan dua tangan dalam pengisyaratan dan memiliki pergerakan yang lebih sederhana tanpa menggunakan imbuhan. Berikut beberapa perbedaan antara BISINDO dan SIBI:

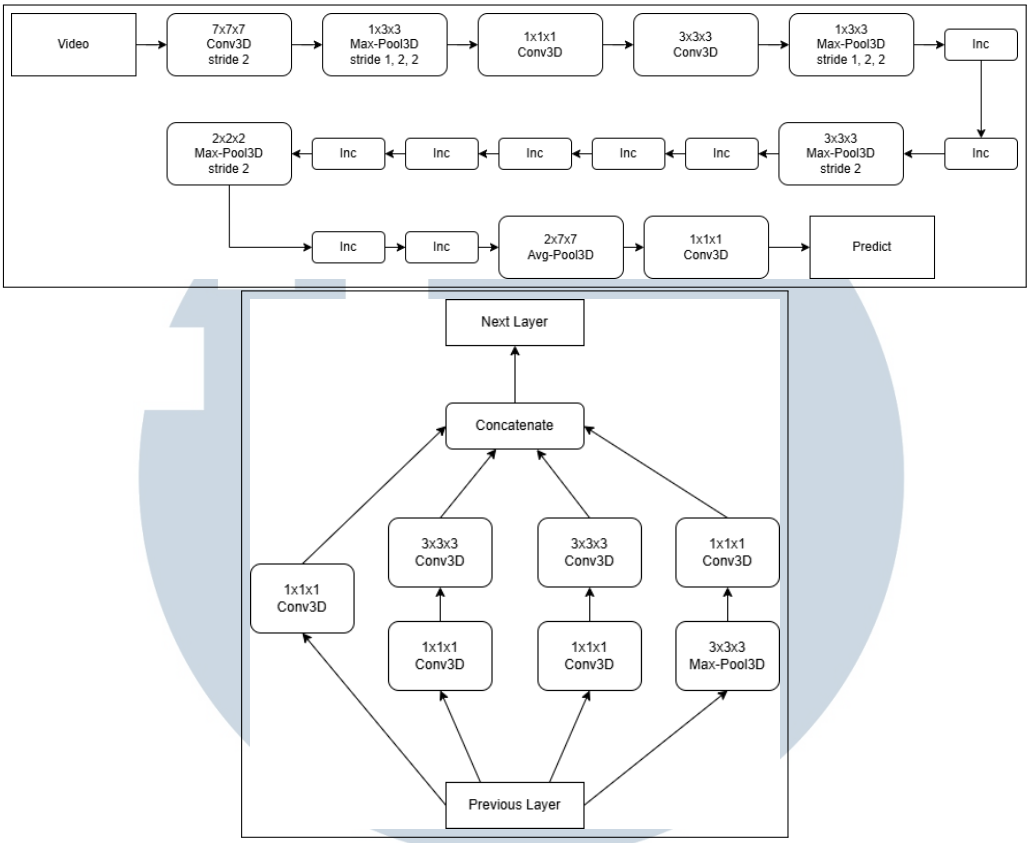
Aspek	SIBI	BISINDO
Asal-usul	Dikembangkan oleh pemerintah	Berkembang secara natural dalam budaya masyarakat Tuli
Tata bahasa	Baku, mengikuti struktur Bahasa Indonesia	Fleksibel, memiliki struktur tersendiri
Penggunaan tangan	Satu tangan	Dua tangan
Konteks penggunaan	Formal, terutama dalam pendidikan	Informal dan komunikasi sehari-hari

Tabel 2.1. Perbandingan Bahasa Isyarat SIBI dan BISINDO

Sumber: [20]

2.2 Inflated 3D (I3D) Model

Inflated 3D (I3D) [21] merupakan *framework* konvolusi jaringan saraf 3D yang didesain untuk menangani prediksi berbasis video. Model ini menggunakan beberapa *Inception* blok dan mengeluarkan prediksi tensor yang berupa array yang berisi probabilitas pada tiap-tiap kelas prediksi. Pada bagian ini akan dijelaskan elemen-elemen yang terdapat dalam I3D secara detil sekaligus dengan infrastrukturnya.

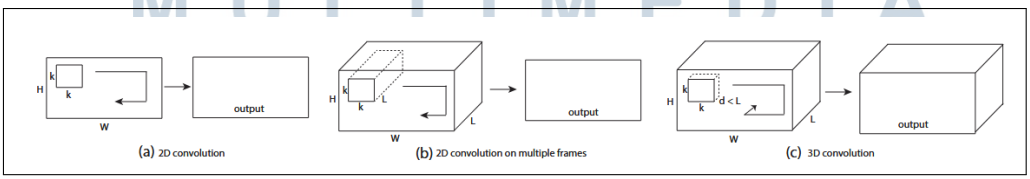


Gambar 2.1. Model framework Inflated 3D (atas) dan Inception module (bawah) pada model Inflated Inception-V1

Sumber: [21]

2.2.1 Jaringan Saraf Konvolusi 3D (Conv3D)

Jaringan saraf konvolusi 3D [22] (Conv3D) adalah sebuah tipe dari jaringan saraf konvolusi yang memproses volumetrik data (contohnya: video, hasil scan medis 3D, atau data sensor spatio-temporal) dengan menggunakan 3D filter (kernel) yang dapat belajar dan mengekstrak fitur dengan mempertahankan hubungan data *spatial* (contoh: gambar) dan *temporal* (contoh: waktu). Metode ini berbeda dengan metode konvolusi 2D yang hanya mengekstrak hubungan spatial pada data.



Gambar 2.2. Cara kerja Conv3D dibandingkan dengan Conv2D

Sumber: [22]

Pada Conv2D [23], konvolusi terjadi pada lapisan konvolusional untuk melakukan ekstraksi fitur berupa *feature maps* pada daerah sekitar menggunakan *feature maps* yang didapat pada lapisan sebelumnya. Kemudian, ditambahkan nilai bias dan hasilnya akan melewati kalkulasi fungsi sigmoid. Secara formal, nilai pada posisi x, y pada *feature map* ke-j di lapisan i dilambangkan sebagai v_{ij}^{xy} melalui fungsi

$$v_{ij}^{xy} = \tanh(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}) \tag{2.1}$$

dimana $\tanh(\cdot)$ merupakan fungsi tangent *hyperbolic*, b_{ij} merupakan nilai bias untuk *feature map* di layer ini, m mengindeks *feature map* di layer ke-$i-1$ yang terkoneksi pada *feature map* yang dipakai, w_{ijk}^{pq} merupakan nilai pada posisi p, q dari kernel yang terhubung ke *feature map* ke-k, dan P_i dan Q_i masing-masing merupakan tinggi dan panjang dari kernel. Pada lapisan *subsampling*, resolusi dari *feature maps* direduksi menggunakan *pooling* daerah sekitar *feature maps* pada lapisan sebelumnya sehingga menghasilkan invariansi untuk maembuat distorsi pada input.

Konvolusi 3D dilakukan dengan melakukan teknik konvolusi menggunakan *kernel* 3D pada input kubik yang dihasilkan dari penumpukan frame-frame yang berurutan. Menggunakan pendekatan ini, *feature maps* keluaran akan terkoneksi ke banyak frame yang berurutan pada lapisan sebelumnya, sehingga mendapatkan informasi pergerakan. Secara formal, nilai *feature map* di lapisan ke-i pada posisi x, y, z dapat dijabarkan sebagai

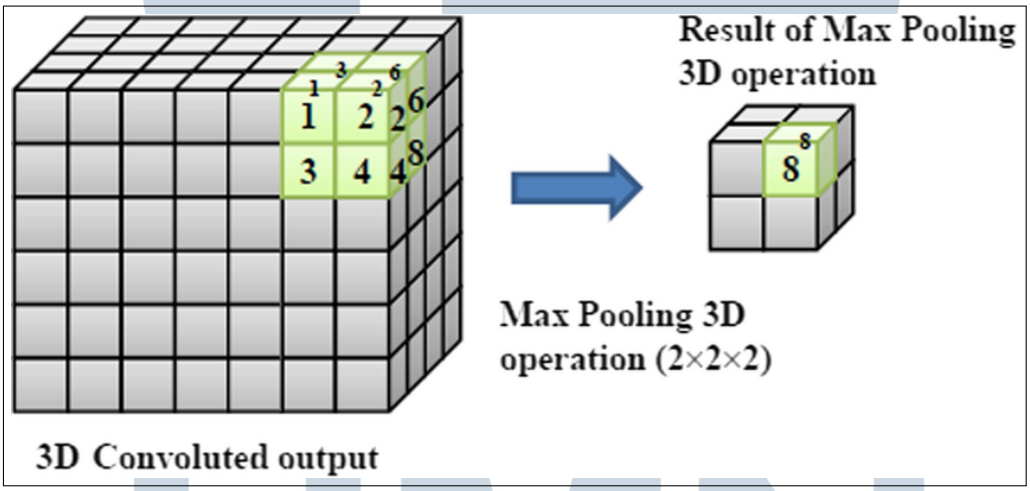
$$v_{ij}^{xyz} = \tanh(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}) \tag{2.2}$$

Dimana R_i merupakan besar dari kernel 3D sepanjang dimensi temporal, w_{ijm}^{pqr} merupakan nilai ke-(p, q, r) dari kernel yang terhubung ke *feature map* ke-m pada lapisan sebelumnya.

Perlu diperhatikan bahwa konvolusi kernel 3 hanya dapat mengekstrak satu tipe fitur dari subset frame kubik, karena berat kernel akan direplikasi melalui seluruh kubik. Maka desain umum dari CNN mempunyai banyak *feature map* yang akan meningkat pada lapisan-lapisan yang sudah mendalam dengan memperbanyak tipe *feature map* dari set dari *feature map* sebelumnya.

2.2.2 3D Max Pooling

Max Pooling merupakan operasi *downsampling* digunakan untuk mengurangi dimensi (kedalaman, tinggi, dan panjang) pada *feature map* yang dikeluarkan oleh Conv3D sambil mempertahankan fitur-fitur yang paling penting. Fitur ini berguna untuk mengurangi kompleksitas *feature map* untuk diberikan ke layer berikutnya sekaligus menjadikan layer berikutnya lebih sensitif terhadap perubahan yang terjadi di input, kemudian Max Pooling juga berguna untuk menghindari model yang overfitting dengan mengurangi banyak parameter pada layer berikutnya. Operasi max pooling menerima parameter input berupa *feature map*, kernel, dan stride.



Gambar 2.3. Mekanisme kerja teknik max pooling 3D

Sumber: [24]

Operasi 3D max pooling menerima parameter berupa input *feature map* berupa tensor 4D Input $\in \mathbb{R}^{(CD_iH_iW_i)}$, parameter besar pooling kernel yang berupa tuple $(K_dK_hK_w)$, stride yang berupa (S_d, S_h, S_w), dan padding opsional (P_d, P_h, P_w) untuk mengontrol dimensi output. Kernel bergerak sesuai dengan stride dan akan melakukan operasi pada setiap pergerakan berupa pengambilan nilai representatif terhadap sebuah subset volume pada input *feature map* yang berupa nilai maksimal di sebuah subset tersebut. Rumus umum dari sebuah operasi pada sebuah subset volume pada *feature map* dapat dijabarkan sebagai berikut:

$$\text{Output}[c,i,j,k] = \max_{m=0}^{K_d-1} \max_{n=0}^{K_h-1} \max_{p=0}^{K_w-1} \text{Input}[c, S_d i+m, S_h j+n, S_w k+p] \tag{2.3}$$

Dimana Output$[c,i,j,k]$ merupakan output max pooling 3D berupa tensor 4D dan $\max_{m=0}^{K_d-1}\max_{n=0}^{K_h-1}\max_{p=0}^{K_w-1}\text{Input}[c,S_d i+m,S_h j+n,S_w k+p]$ merupakan nilai maksimum pada sebuah subset volume input berdasarkan ukuran kernel. Bentuk output max pooling 3D akan lebih rendah dari bentuk input dan dipengaruhi oleh besar kernel, stride, dan padding. Operasi ini akan memberikan output Output $\in \mathbb{R}^{(CD_oH_oW_o)}$ dimana nilai pada masing-masing dimensi output dapat dijabarkan menggunakan rumus:

$$D_o = \lfloor (D_i + 2 * P_d - K_d)/S_d \rfloor + 1 \tag{2.4}$$

$$H_o = \lfloor (H_i + 2 * P_h - K_h)/S_h \rfloor + 1 \tag{2.5}$$

$$W_o = \lfloor (W_i + 2 * P_w - K_w)/S_w \rfloor + 1 \tag{2.6}$$

Sementara nilai dimensi C yang berupa dimensi channel tidak berubah

2.2.3 Inception V1

Inception V1 [25] merupakan arsitektur jaringan konvolusional yang digunakan untuk melakukan deteksi gambar dan merupakan *baseline* dari jaringan I3D. Ciri khas dari arsitektur ini adalah dengan menggunakan banyak modul *Inception* yang menggantikan modul konvolusi langsung.

A Modul *Inception*

Inspirasi penggunaan modul *Inception* diawali pada penelitian [26] dimana pada penelitian ini ingin mengatasi permasalahan klasik pada model jaringan konvolusi yang berupa penambahan *layer* konvolusional akan membuat model mendeteksi gambar secara lebih mendetail, namun disisi lain akan membuat model lebih rentan terhadap *overfitting* dikarenakan kompleksitas model berupa jumlah parameter yang lebih tinggi. Oleh karena itu, daripada menambah sebuah *layer* konvolusi, penelitian ini menggunakan modul pencabangan dari sebuah layer yang dinamakan modul *Inception*.

Modul inception membagi *layer* sebelumnya menjadi 4 cabang utama dan mempunyai kedalaman 2 *layer* yang kemudian akan dikonkatenasi pada akhir modul menjadi 1 *feature map*. *Layer* pertama berupa konvolusi menggunakan dimensi kernel 1x1 dan max-pooling 3x3, *layer* ini berfungsi untuk mengurangi

dimensi berupa jumlah parameter pada *layer* input sebelumnya. Kemudian dilanjutkan ke *layer* kedua yang berupa konvolusi 3x3 dan konvolusi 1x1 yang berfungsi untuk melakukan prediksi. Hasil dari modul *Inception* adalah modul yang dapat mendeteksi data spatial yang lebih mendetail namun mempunyai parameter yang sebanding ketimbang menggunakan satu proses konvolusi.

B Arsitektur Inception V1

Arsitektur Inception V1 didesain dengan inspirasi dari jaringan Le Net 5 [27] dengan mempertimbangkan efisiensi komputasi [25] sehingga model dapat dijalankan pada perangkat yang bersifat terbatas. Jaringan terdiri dari 22 *layer* konvolusi (atau 27 layer jika *layer* pooling dihitung), terdapat sekitar 100 blok modul yang digunakan tergantung dari sistem *machine learning* yang digunakan. Penggunaan *Average Pooling* sebelum klasifikasi dilakukan berdasarkan [26] dan mendapatkan peningkatan akurasi ketimbang tidak menggunakan modul tersebut. Di akhir arsitektur terdapat penggunakan modul Dropout dengan rasio sebesar 70% untuk pencegahan overfitting.

2.2.4 Arsitektur I3D

Arsitektur I3D[21] merupakan modifikasi dari arsitektur Inception V1 [25] untuk dapat beradaptasi pada dimensi waktu di sebuah video. Modifikasi yang dilakukan adalah dengan menambah dimensi temporal (waktu) dengan mengubah metode konvolusi yang sebelumnya bersifat 2D menjadi 3D (contoh: Conv2D menjadi Conv3D dan Max-Pooling2D menjadi Max-Pooling3D) *inflated*, sehingga modifikasi ini tidak mengubah arsitektur dasar. Arsitektur I3D meliputi 3 bagian utama: layer input, layer *inception*, dan layer klasifikasi.

Layer input merupakan layer yang menerima input berupa matriks tensor yang berdimensi Input Tensor :$\in \mathbb{R}^{(BTHWC)}$ dimana B merupakan jumlah batch, T merupakan potongan temporal (contoh: 64 frame pada video), H dan W merupakan resolusi spatial (contoh: 224x224 video), dan C merupakan banyak channel. Layer *inception* adalah bagian dimana model berlatih berdasarkan data dari layer input. Layer ini meliputi *konvolusi 3D*, *max pooling 3D*, dan Inception module. Inception module membagi satu input dari layer sebelumnya dan diberikan ke 4 aliran konvolusi dan kemudian digabung menjadi 1 dan diberikan ke layer berikutnya. Layer terakhir adalah layer output yang terdiri dari tensor array skor klasifikasi

setiap *class* menggunakan metode aktivasi *softmax*.

2.3 Algoritma aliran optik

Algoritma aliran optik merupakan algoritma yang digunakan untuk mendeteksi pergerakan di dalam sebuah video dengan membandingkan sebuah frame dalam video dengan frame sebelumnya. Algoritma ini menerima input berupa gambar yang berurutan dari sebuah video. Kemudian, algoritma ini menghasilkan output berupa 2 channel vektor (x, y) yang berisi skor besaran pergerakan pada setiap piksel pada setiap frame di sebuah video. Terdapat berbagai macam algoritma yang dapat digunakan untuk mengkalkulasi aliran optik. Penelitian ini akan menjelaskan secara detail tentang dua algoritma aliran optik yang akan digunakan, yaitu TV-L1 dan Farneback.

Dasar teori aliran optik mengasumsikan bahwa intensitas piksel dalam suatu objek tidak berubah terhadap waktu meskipun objek tersebut bergerak. Asumsi ini dikenal sebagai *brightness constancy constraint*. Anggap sebuah objek di waktu t dengan intensitas $I(x,y,t)$, setelah waktu dt, intensitas objek tersebut menjadi $I(x+dx, y+dy, t+dt)$. Diasumsikan bahwa intensitas pada waktu t dan $t+dt$, sehingga mendapatkan persamaan:

$$I(x,y,t) = I(x+dx, y+dy, t+dt) \tag{2.7}$$

Dengan pendekatan deret Taylor dan mengabaikan orde lebih tinggi, diperoleh bentuk linear:

$$\frac{dI}{dx}\delta x + \frac{dI}{dy}\delta y + \frac{dI}{dt}\delta t = 0 \tag{2.8}$$

Kemudian dengan membagi kedua persamaan dengan δt, persamaan aliran optik dapat didapat:

$$\frac{dI}{dx}u + \frac{dI}{dy}v + \frac{dI}{dt} = 0 \tag{2.9}$$

dengan u adalah dx/dt dan v adalah dy/dt. Kemudian $\frac{dI}{dx}$ merupakan gradien terhadap axis horizontal, $\frac{dI}{dy}$ merupakan gradien terhadap axis vertikal, dan $\frac{dI}{dt}$ merupakan gradien terhadap waktu. Tugas dari algoritma-algoritma aliran optik

adalah dengan mencari nilai gradien u dan v.

2.3.1 *Total Variation regularization with L1 data fidelity* (TV-L1)

Metode TV-L1 memakai kondisi *brightness constancy constraint* menggunakan persamaan

$$\nabla I \cdot + \frac{\partial}{\partial t} I = 0 \tag{2.10}$$

Untuk semua point di dalam gambar, kondisi $\nabla I \cdot + \frac{\partial}{\partial t} I = 0$ merupakan persamaan linear pada dua variabel (komponen u yang berupa koordinat $u(x,y)$). Oleh karena itu, ada dua variabel namun mempunyai satu persamaan sehingga persamaan tidak dapat ditentukan. Solusi standar yang dapat dilakukan adalah dengan memakai *smoothness condition* dimana u dapat diregularisasi seperti proposal Horn dan Schnuck [28] untuk meminimalisir nilai u:

$$E_{\text{Horn\_Schnuck}}(u) = \int_{\Omega} (\nabla I \cdot u + \frac{\partial}{\partial t} I)^2 + \alpha(|\nabla u_1|^2 + |\nabla u_2|^2) \tag{2.11}$$

Teknik minimalisasi ini mudah diselesaikan dengan menggunakan metode standar. Pengurangan nilai menggunakan $|\nabla u_1|^2 + |\nabla u_2|^2$ mengubah gradien u yang tinggi sehingga mengurangi ketidaktentuan persamaan. Persamaan ini laizmnya akan diubah menggunakan formulasi non-linear $I_1(x+u) - I_0(x) = 0$ untuk urutan gambar, formulasi pada $I_1(x+u)$ dapat dilinearkan menggunakan ekspansi Taylor, sehingga menjadikan persamaan

$$\rho(u) = \nabla I_1(x+u^0) \cdot (u-u^0) + I_1(x+u^0) - I_0(x) = 0 \tag{2.12}$$

Dengan u^0 merupakan estimasi nilai u. persamaan Horn-Schnunck bisa dimodifikasi lebih lanjut sehingga mengizinkan diskontinuitas pada bidang aliran dengan mengubah faktor kuadrat menggunakan algoritma yang meminimalisir fungsi energi yaitu jumlah dari variasi u dan istilah L^1

$$E(u) = \int_{\Omega} |\nabla u_1| + |\nabla u_2| + \lambda |\rho(u)| \tag{2.13}$$

Cara untuk meminimalisir energi secara efisien dilakukan dengan merelaksasi konveks

$$E_0(u,v) = \int_\Omega |\nabla u_1| + |\nabla u_2| + \frac{1}{2\theta}|u-v|^2 + \lambda|\rho(v)| \tag{2.14}$$

Settingan nilai θ menjadi sangat kecil memaksa nilai E_0 minimum untuk terjadi ketika nilai u dan v hampir sama, sehingga mengurangi energi original E yang didefinisikan pada persamaan 2.14. Yang diperhatikan disini merupakan relaksasi dari E_0 yang dapat diminimalisir dengan menentukan satu nilai u atau v dan menyelesaikan variable yang lain

Jika nilai v ditentukan, kerjakan

$$\min_u \int_\Omega |\nabla u_1| + |\nabla u_2| + \frac{1}{2\theta}|u-v|^2 \tag{2.15}$$

Jika nilai u ditentukan, kerjakan

$$\min_v \int_\Omega \frac{1}{2\theta}|u-v|^2 + \lambda|\rho(v)| \tag{2.16}$$

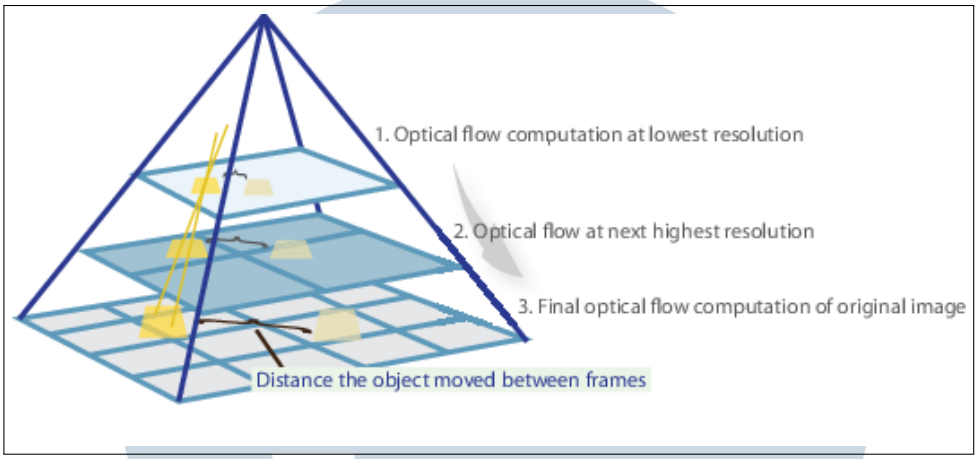
Persamaan pertama cocok engan model Rudin-Osher-Fatemi [29] terkait dengan algoritma model *total variation denoising* yang dapat diselesaikan dengan menggunakan algoritma *duality-based* yang ditemukan oleh Chambolle [30]. Sedangkan persamaan kedua dapat diselesaikan secara poin menggunakan ambang batas nilai v dikarenakan persamaan tidak bergantung kepada turunan v.

Algoritma TV-L1 dapat dipisah menjadi dua modul untuk memecahkan masalah pengambilan nilai u dan v, yaitu prosedur untuk kalkulasi aliran optik pada skala yang ditetapkan dan algoritma utama yang melakukan implementasi skema piramida untuk mendapatkan perkiraan solusi.

2.3.2 Algoritma aliran optik Farneback

Algoritma Farnebäck [15] merupakan salah satu metode aliran optik yang dikembangkan oleh Gunnar Farnebäck pada tahun 2003. Metode ini bertujuan untuk menghitung vektor gerakan untuk setiap piksel antar dua frame gambar.

Tidak seperti metode TV-L1 metode Farnebäck menghasilkan *dense flow field*, yaitu pergerakan pada seluruh piksel.



Gambar 2.4. Visualisasi algoritma farneback
Sumber: [31]

Metode ini menggunakan pendekatan bahwa sebuah titik pada gambar dapat direpresentasikan sebagai fungsi kuadrat (*second-order polynomial*):

$$f(\mathbf{x}) \sim \mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}^T \mathbf{x} + c \tag{2.17}$$

dimana:

- $\mathbf{x} = (x, y)^T$: koordinat piksel pada waktu T
- $\mathbf{A}$: matriks simetris 2×2,
- $\mathbf{b}$: vektor gradien,
- c: konstanta intensitas.

Jika suatu titik $f(\mathbf{x})$ pada frame pertama berpindah sejauh $\Delta\mathbf{d}$ pada frame kedua, maka bagian baru dapat dituliskan sebagai:

$$f_2(\mathbf{x}) \approx f_1(\mathbf{x} + \mathbf{d}) = (\mathbf{x} + \mathbf{d})^T \mathbf{A}(\mathbf{x} + \mathbf{d}) + \mathbf{b}^T(\mathbf{x} + \mathbf{d}) + c \tag{2.18}$$

Dengan merekonstruksi dua titik yang berurutan dan menghitung transformasi antar keduanya, kita dapat mengestimasi vektor pergerakan $\mathbf{d}$.

2.3.3 Estimasi Gerakan

Untuk memperoleh **d**, Farnebäck melakukan:

1. Aproksimasi pergerakan gambar lokal dengan fungsi kuadrat di setiap piksel.
2. Pencocokan dua patch kuadrat pada dua frame.
3. Estimasi perbedaan posisi (*displacement vector*) dengan meminimalkan perbedaan fungsi kuadrat tersebut.

Proses ini dilakukan secara *multiscale (pyramid)* untuk menangkap gerakan besar dan halus.

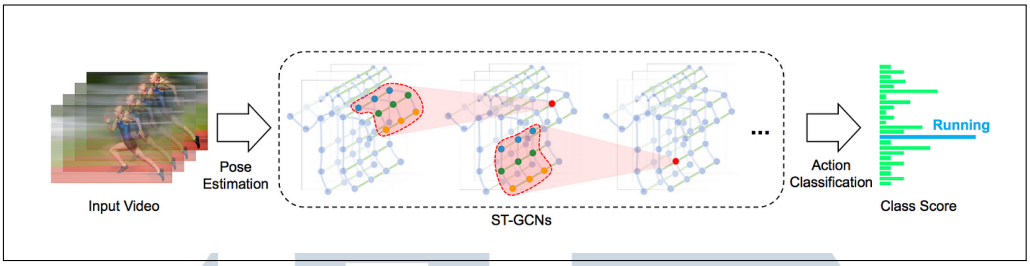
2.3.4 Langkah-Langkah Algoritma

Langkah utama dalam metode Farnebäck adalah:

1. Pembangunan piramida (representasi skala).
2. Pada level piramida teratas (resolusi terkecil), aproksimasi gambar sebagai fungsi kuadrat.
3. Estimasi vektor gerak **d** dengan mencari transformasi antar titik.
4. perhalus estimasi ke level lebih rendah dengan metoda warping dan interpolasi.
5. Proses diulangi sampai mencapai resolusi penuh.

2.4 Spatial Temporal Graph Convolutional Network (ST-GCN)

ST-GCN [32] merupakan model pembelajaran mendalam konvolusi berbasis graf yang dibuat untuk mempelajari pergerakan manusia melalui sambungan sendi dan titik-titik koordinat (2D atau 3D) yang bergerak berdasarkan waktu (contohnya pergerakan titik-titik pose pada sebuah video). Berbeda dengan model CNN atau RNN yang kurang cocok untuk menangani data graf, ST-GCN menggunakan konvolusi graf untuk memproses data spatial seperti struktur tubuh dan temporal seperti pergerakan dari waktu-ke-waktu secara bersamaan.

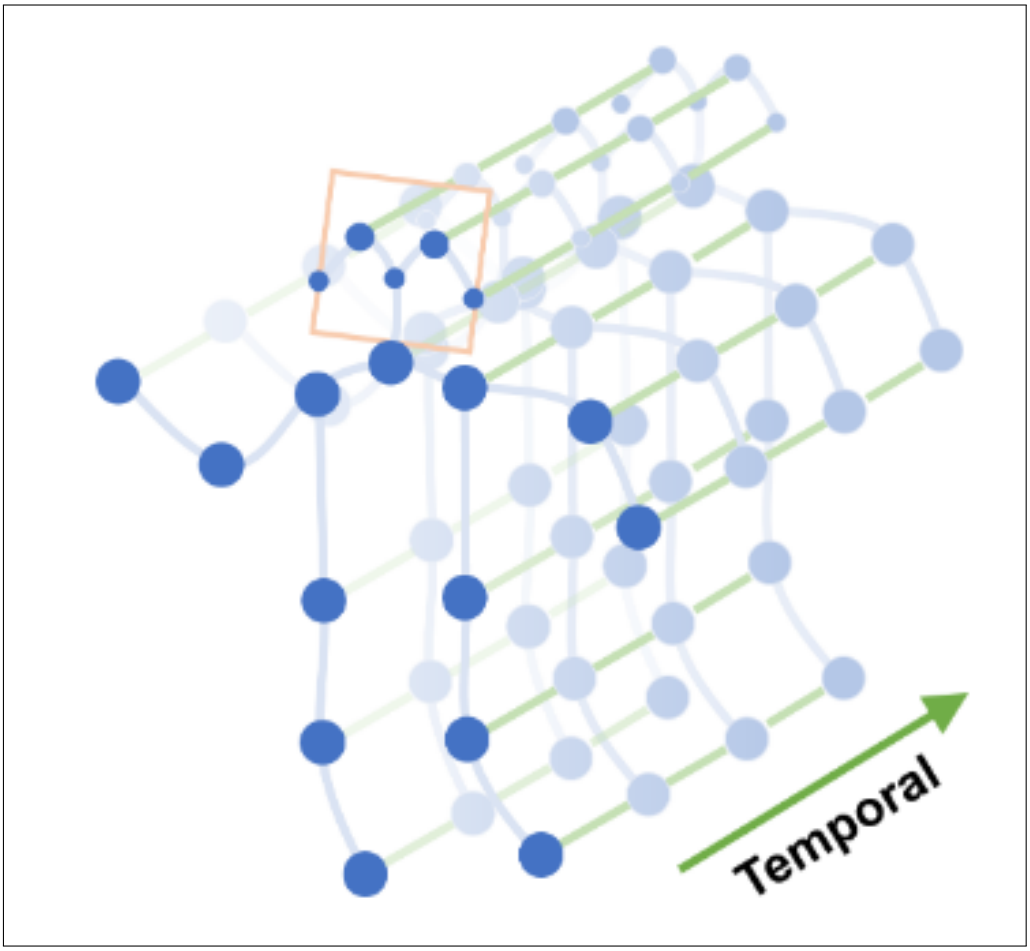


Gambar 2.5. Mekanisme kerja teknik konvolusi pada operasi ST-GCN

Sumber: [32]

2.4.1 Konstruksi Graf Pose Tubuh

Pose tubuh manusia biasanya direpresentasikan dalam koordinat 2D atau 3D pada setiap sendi pada setiap waktu. ST-GCN mengkonstruksikan graf spatial-temporal tanpa arah $G = (V, E)$ pada sebuah urutan pose manusia di sebuah video dengan N banyak sendi dan jumlah frame T yang melibatkan koneksi antar-sendi dan antar frame.



Gambar 2.6. Koneksi antar sendi manusia yang berurutan terhadap waktu di ST-GCN

Sumber: [32]

Pada graf ini, set node $V = \{v_{ti} | t = 1,...,T, i = 1,...,N\}$ mencakup seluruh sendi di sebuah urutan pose. Inputan dari ST-GCN berupa koordinat vektor berupa node $F(v_{ti})$ dan estimasi kepercayaan pada sendi ke-i di frame ke-t. Konstruksi dari graf terdiri dari 2 tahapan. Pertama-tama, sendi-sendi dari sebuah frame akan terkoneksi terhadap koneksi yang terdapat pada struktur tubuh manusia seperti pada gambar 2.6. Kemudian, setiap sendi akan terkoneksi pada sendi-sendi yang sama pada frame berikutnya. Konektivitas pada setiap sendi dapat ditulis sebagai 2 subset himpunan $E_S = \{v_{ti}v_{tj} | (i, j) \in H\}$, dimana H adalah set dari koneksi sendi di pose tubuh. Kemudian $E_F = \{v_{ti}v_{(t+1)i}\}$ yang berupa konektivitas sendi antar waktu, sehingga semua koneksi pada sendi ke-i di E_F merepresentasikan pergerakan terhadap waktu.

2.4.2 Konvolusi Graf Spatial Temporal

Pada model graf CNN pada sebuah frame di waktu τ, terdapat N-buah node sendi V_t sepanjang koneksi antar sendi $E_S(\tau) = \{v_{ti}v_{tj}|t = \tau, (i,j) \in H\}$. Pada operasi konvolusi gambar 2D atau *feature map*, output *feature map* masih berdimensi 2D. Dengan nilai *stride* 1 dan *padding*, output dari konvolusi bisa mempunyai ukuran yang sama dengan inputnya. Menggunakan kondisi ini, jika terdapat operasi konvolusi dengan jumlah *kernel* $K \times K$ dan output *feature map* f_{in} dengan banyak channel c, maka output pada sebuah channel di lokasi spatial x dapat ditulis sebagai:

$$f_{out}(x) = \sum_{h=1}^{K} \sum_{w=1}^{K} f_{in}(p(x,h,w)) \cdot w(h,w) \tag{2.19}$$

Dimana fungsi sampling $p : Z^2 \times Z^2 \rightarrow Z^2$ menghitung tetangga pada lokasi x. fungsi berat $w : Z^2 \rightarrow \mathbb{R}^c$ menyediakan vektor berat pada dimensi-c untuk komputasi keluaran dari fitur input yang berdimensi c. Operasi konvolusi dalam graf dilakukan dengan menambah formulasi dari kasus di atas dimana input *feature map* terdiri dari graf V_t. Sehingga *feature map* $f_{in}^t : V_t \rightarrow R^c$ punya sebuah vektor pada setiap node di dalam graf. ST-GCN juga mendefinisikan ulang fungsi sampling p dan fungsi berat w

Pada fungsi sampling pada gambar $p(h,w)$ didefinisikan pada piksel tetanganya terhadap lokasi tengah x. Sedangkan pada grraf dapat didefinisikan secara mirip dengan konektivitas antar tetangga $B(v_{ti} = \{v_{tj}|d(v_{tj}, v_{ti}) \leq D\}$ dari node v_{ti}. Disini, $d(v_{tj}, v_{ti})$ berupa panjang minimum dari sebuah hubungan v_{tj} ke v_{ti}. Sehingga fungsi sampling $p : B(v_{ti} \rightarrow V$ dapat ditulis menjadi

$$p(v_{tj}, v_{ti}) = v_{tj} \tag{2.20}$$

Untuk fungsi berat, perubahan lebih sulit dilakukan dikarenakan lokasi tetangga dari sebuah titik tengah sudah terdapat secara alami di konvolusi 2D. ST-GCN memakai teknik yang diterapkan pada usulan sebelumnya [33] dimana urutan pada graf di-label berdasarkan tetangganya. Namun, daripada memberikan setiap node label yang unik, dapat disimplifikasikan dengan melakukan partisi pada setiap set tetanga $B(v_{ti})$ dari node sendi v_{ti} menjadi jumlah subset K yang tetap dimana setiap subset mempunyai label numerik. Sehingga dapat dilakukan pemetaan

$I_{ti} : B(v_{ti}) \rightarrow \{0, ..., K-1\}$ yang memetakan node di tetangga ke masing-masing subset labelnya. Sehingga fungsi berat $w(v_{ti}, v_{tj}) : B(v_{ti} \rightarrow R^c$ dapat diimplementasi dengan mengindeks tensor dari dimensi (c, K) atau

$$w(v_{ti}, v_{tj}) = w'(l_{ti}(v_{tj})) \tag{2.21}$$

Dengan fungsi sampling dan berat yang telah didefinisikan ulang, maka rumus konvolusi pada persamaan 2.22 dapat ditulis ulang menjadi:

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(p(v_{ti}, v_{tj})) \cdot w(v_{ti}, v_{tj}) \tag{2.22}$$

Dimana normalisasi $Z_{ti}(v_{tj}) = |\{v_{tk} | l_{ti}(v_{tk}) = l_{ti}(v_{tj})\}|$ sama dengan nilai kardinalitas dari subset koresponden yang digunakan untuk menyeimbangkan kontribusi pada setiap subset ke output. Dengan substitutsi pada persamaan sebelumnya, maka fungsi dapat ditulis menjadi

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj})) \cdot w(l_{ti}(v_{tj})) \tag{2.23}$$

Seangkan untuk mengintroduksi dimensi temporal dilakukan dengan mengkoneksikan sendi yang sama pada frame yang berurutan. Sehingga koneksi dilakukan dengan memakai konsep tetangga, sehingga menjadi

$$B(v_{ti}) = \{v_{qj} | d(v_{tj}, v_{ti}) \leq K, |q-t| \leq \lfloor \Gamma/2 \rfloor\} \tag{2.24}$$

Dimana parameter Γ mengontrol panjang temporal untuk dimasukkan ke perhitungan konvolusi, sehingga dinamakan *temporal kernel size*. Untuk menyelesaikan operasi konvolusi pada graf spatial temporal, dibutuhkan pula fungsi sampling dan berat yang sama seperti perhitungan spatial. Karena urutan axis temporal berbentuk padu, dilakukan modifikasi *label map* l_{ST} untuk tetangga spatial temporal yang berakar pada v_{ti}

$$l_{ST}(v_{qj}) = l_{ti}(v_{tj}) + (q - t + \lfloor \Gamma/2 \rfloor) \times K \tag{2.25}$$

dimana $l_{ti}(v_{tj})$ merupakan *label map* untuk kasus satu frame pada v_{ti}.

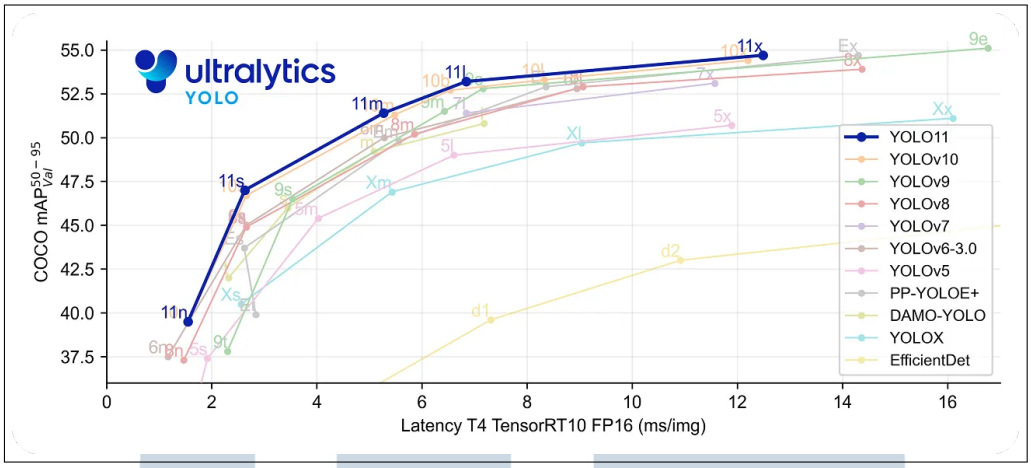
2.5 You Only Look Once (YOLO)

YOLO [17] merupakan *framework* model konvolusional yang cepat dan akurat yang digunakan untuk mendeteksi objek. YOLOv11 merupakan peningkatan dari versi sebelumnya, berikut merupakan tabel perbandingan antara versi-versi YOLO:

Versi	Tahun	Tasks	Kontribusi	Framework
YOLO [34]	2015	Deteksi Objek, Klasifikasi	Model deteksi objek	Darknet
YOLOv2 [35]	2016	Deteksi Objek, peningkatan klasifikasi	Multi-Skala Training, pengelompokan dimensi	Darknet
YOLOv3 [18]	2018	Deteksi Objek, Deteksi Multi-Skala	SPP blok, *backbone* Darknet-53	Darknet
YOLOv4 [36]	2020	Deteksi Objek, *tracking* objek	aktivasi Mish, *backbone* CSPDarknet-53	Darknet
YOLOv5 [37]	2020	Deteksi Objek, segmentasi instansi	deteksi *anchor-free*, aktivasi SWISH, PANet	PyTorch
YOLOv6 [38]	2022	Deteksi Objek, segmentasi instansi	*self-attention*, deteksi objek *anchor-free*	PyTorch
YOLOv7 [39]	2022	Deteksi Objek, tracking objek, segmentasi instansi	*Transformers*, E-ELAN reparameterisasi	PyTorch
YOLOv8 [40]	2023	Deteksi Objek, segmentasi instansi, segmentasi panoptik, estimasi *keypoint*	GAN, deteksi *anchor-free*	PyTorch
YOLOv9 [41]	2024	Deteksi Objek, segmentasi instansi	PGI, GELAN	PyTorch
YOLOv10 [42]	2024	Deteksi Objek	*NMS-free training*	PyTorch

Tabel 2.2. Versi-versi YOLO terdahulu

Sumber: [17]



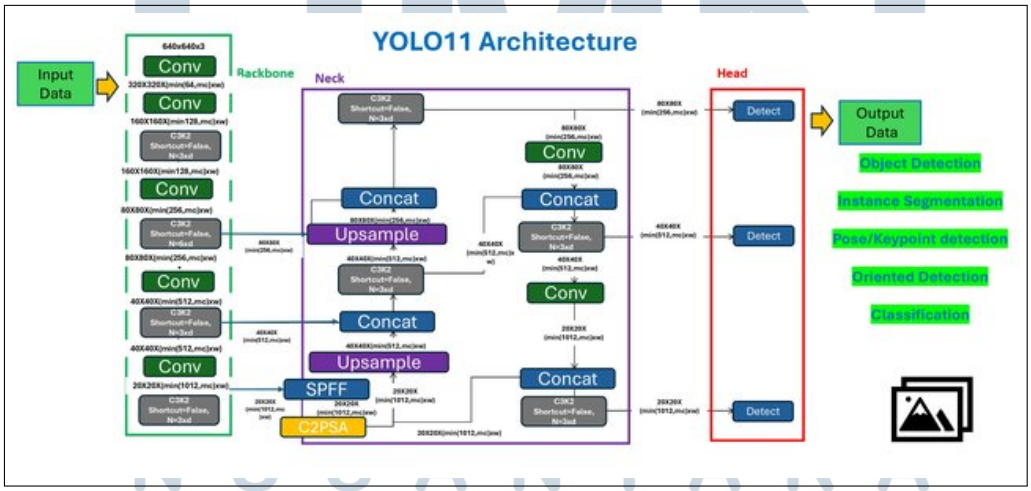
Gambar 2.7. Perbandingan akurasi antara YOLOv11 dengan versi-versi YOLO sebelumnya

Sumber: [43]

Sedangkan YOLOv11 [17] merupakan peningkatan dari YOLOv10 dan merupakan iterasi terkini.

2.5.1 Arsitektur YOLOv11

Arsitektur YOLOv11 terddiri dari 3 komponen utama, yaitu *backbone* yang berfungsi sebagai ekstraksi fitur untuk mengubah data gambar ke *feature maps*, kemudian *neck* sebagai teknik pemrosesan, dan terakhir *head* sebagai mekanisme prediksi untuk mengeluarkan output final.



Gambar 2.8. Arsitektur YOLOv11

Sumber: [44]

2.5.2 Backbone

A Lapisan Konvolusi

Struktur YOLOv11 mirip seperti versi sebelumnya, menggunakan lapisan konvolusi untuk melakukan *downsampling* terhadap gambar. Lapisan-lapisan ini membentuk fondasi dari proses ekstraksi fitur. Perubahan yang dilakukan YOLOv11 adalah dengan memperkenalkan blok C3k2 sebagai pengganti blok C2f [17]. Blok C3k2 merupakan implementasi *Cross Stage Partial* (CSP) *Bottleneck* yang lebih efisien dengan menggunakan 2 konvolusi yang lebih rendah dibandingkan satu konvolusi besar. ”k2” di C3k2 melambangkan ukuran kernel yang lebih rendah, sehingga berkontribusi ke pemrosesan gambar yang lebih cepat sekaligus menjaga performa.

B SPPF dan C2PSA

YOLOv11 masih memakai blok *Spatial Pyramid Pooling - Fast* (SPPF) yang dipakai pada versi sebelumnya [17]. Namun, memakai blok *Cross Stage Partial with Spatial Attention* (C2PSA) setelah SPPF. C2PSA merupakan blok yang digunakan untuk meningkatkan attensi spatial pada *feature maps* sehingga membuat model dapat berfokus ke area yang penting dalam sebuah gambar.

2.5.3 Neck

A Blok C3k2

YOLOv11 mengubah blok C2f di *neck* menjadi blok C3k2 agar dapat lebih cepat dan efisien sehingga performa keseluruhan menjadi lebih baik.

B Mekanisme Atensi

YOLOv11 menambah fokus atensi spatial dengan menggunakan modul C2PSA untuk berkonsentrasi pada area yang dicari pada gambar. Sehingga mempunyai potensi untuk memberikan deteksi yang lebih akurat.

2.5.4 Head

A Blok C3k2

Pada bagian *head*, YOLOv11 menggunakan beberapa blok C3k2 untuk memproses dan mengekstrak *feature maps* dengan lebih efisien. Blok C3k2 ditaruh pada beberapa aliran pada *head* sehingga memungkinkan pemrosesan pada gambar dengan berbagai skala dan kedalaman yang berbeda. Blok C3k2 bekerja secara fleksibel tergantung kepada parameter C3k2:

- ketika c3k = False, modul C3k2 bekerja mirip seperti C2f
- ketika c3k = True, struktur *bottleneck* diganti menjadi modul C3 sehingga ekstraksi fitur dapat dilakukan secara mendalam dan lebih kompleks

2.5.5 Blok CBS

Head pada YOLOv11 menggunkan beberapa blok *Convolution-BatchNorm-Silu* (CBS) setelah blok C3k2. Blok ini memproses *feature maps* dengan:

- melakukan ekstraksi pada fitur-fitur yang relevan untuk deteksi objek yang lebih akurat
- Melakukan *batch normalization* untuk normalisasi dan menstabilkan data
- menggunakan fungsi aktivasi *Sigmoid Linear Unit* (SiLU) untuk nonlinearitas sehingga performa model bertambah

2.5.6 Lapisan Konvolusi Akhir dan Lapisan Deteksi

Setiap cabang deteksi diakhiri dengan lapisan Conv2D untuk mengurangi fitur menjadi koordinat dan prediksi kelas. Prediksi akhir terdiri dari:

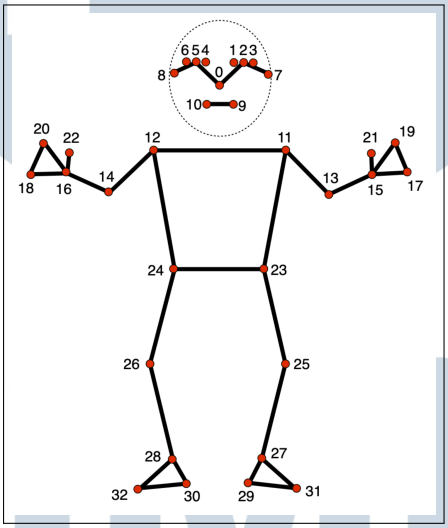
- *Bounding box* berupa kotak koordinat lokal dimana objek berada
- Skor kepercayaan sebuah objek berada
- Skor kelas berupa tipe objek yang terdeteksi

2.6 MediaPipe

MediaPipe [12] merupakan sebuah *library* yang digunakan untuk melakukan deteksi objek dan estimasi pose tubuh. *Library* ini bersifat *open-source* sehingga dapat dimodifikasi oleh publik. Bagian ini akan menjelaskan tentang fitur-fitur yang dapat digunakan pada MediaPipe terkait pada penelitian.

2.6.1 Estimasi Pose Tubuh

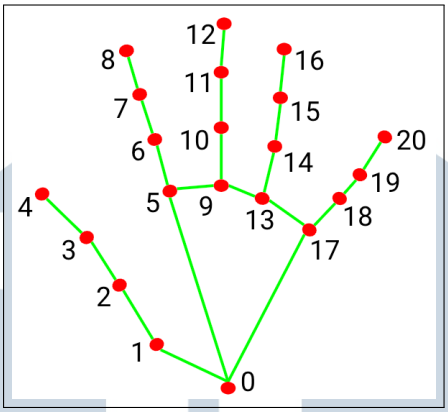
MediaPipe dapat melakukan estimasi pose tubuh [14] berupa titik-titik koordinat 3-dimensi (xyz) yang telah dinormalisasi berdasarkan skala dimensi pada sebuah gambar. Titik-titik pose terdiri dari 33 lokasi yang terdiri dari bagian tubuh, kepala, dan tangan.



Gambar 2.9. Titik-titik pose tubuh yang diprediksi oleh MediaPipe
Sumber: [14]

2.6.2 Estimasi Pose Tangan

MediaPipe juga dapat mengestimas pose tangan [45] yang berupa 20 titik-titik koordinat 3 dimensi (xyz) yang telah dinormalisasi pada sebuah gambar.



Gambar 2.10. Titik-titik pose tangan yang diprediksi oleh MediaPipe

Sumber: [46]