BAB II
LANDASAN TEORI

Berikut ini merupakan beberapa teori yang digunakan dalam penelitian yang dilakukan.

2.1. Tinjauan Pustaka

Dalam penelitian ini ada beberapa pustaka yang saling berhubungan dan mendukung pada kegiatan penelitian ini, antara lain :

(Chen, Y., Tao, G., Ren, H., Lin, X., & Zhang, L., 2018), Penelitian ini yang berjudul *Accurate seat belt detection in road surveillance images based on CNN and SVM.* Penelitian ini dilakukan untuk Mendeteksi objek sabuk pengaman pada citra gambar. Tujuan penelitian ini adalah untuk mengukur keakurasian dalam mendeteksi objek sabuk pengaman pada citra gambar. Kontribusi penelitian ini adalah mengetahui proses deteksi sabuk pengaman pada citra gambar dengan menerapakan metode *Convolution Neural Network* dan *Support Vector Machine*.

(Dewi, Kristiana, Darlis, & Dwiputra, 2019), Penelitian ini yang berjudul *Deep Learning RetinaNet based Car Detection for Smart Transportation Network.* Penelitian ini berfokus pada *object detection* atau *car detection* sebagai objek yang dideteksi. Objek yang dideteksi menampilkan bounding box, label kelas dan nilai klasifikasi. Tujuan penelitian ini adalah mengukur keakurasian metode CNN dengan model *RetinaNet* untuk mendeteksi objek mobil. Kontribusi penelitian ini adalah mengetahui proses deteksi objek mobil dengan metode CNN dengan model *RetinaNet*.

(Lubis & Suprin, 2018), Penelitian ini berjudul Perancangan Aplikasi untuk Mendeteksi Sabuk Pengaman Mobil Menggunakan Algoritma *Backpropagation Neural Network.* Penelitian ini dilakukan untuk mendeteksi penggunaan sabuk pengaman dengan cara melakukan proses akusisi citra dan menggunakan metode *Backpropagation Neural Network* (BPNN). Tujuan penelitian ini adalah untuk mendeteksi pengemudi yang menggunakan sabuk pengaman dan tidak menggunakan sabuk pengaman dengan metode *Backpropagation Neural Network*. Kontribusi penelitian ini adalah mengetahui proses deteksi mendeteksi pengemudi yang

menggunakan sabuk pengaman dan tidak menggunakan sabuk pengaman dengan metode *Backpropagation Neural Network.*

(Tianshu & Zhijia, 2019), Penelitian ini berjudul *Detection and Implementation of Driver's Seatbelt Based on FPGA*. Penelitian ini dilakukan untuk mendeteksi pengemudi mobil yang tidak menggunakan sabuk pengaman dengan metode *Yolo V1*. Tujuan penelitian ini untuk karakteristik perangkat keras dari FPGA, algoritma *Yolo* digunakan untuk mendeteksi pengemudi yang tidak menggunakan sabuk pengaman dan direalisasikan oleh *IP core* dari *open source* XILINX. Percepatan algoritma diwujudkan dengan memanggil perangkat keras FPGA di sisi ARM. Kontribusi penelitian ini adalah mengetahui proses untuk mendeteksi pengemudi yang tidak menggunakan sabuk pengaman dengan metode *Yolo V1*.

(Chun, 2019), penelitian ini berjudul *NADS-Net: A Nimble Architecture for Driver and Seat Belt Detection via Convolutional Neural Networks*. Penelitian ini dilakukan untuk mendeteksi pengemudi dan sabuk pengaman dengan Model *NADS-Net* dengan arsitektur *ResNet*-50. Proses yang dilakukan untuk mendeteksi pengemudi dan sabuk pengaman meliputi proses *konvolusi, Feature Pyramid Network,* dan *ekstraksi feature map*. Tujuan penelitian ini adalah untuk mengukur keakurasian dalam mendeteksi pengemudi dan sabuk pengaman dengan model *NADS-Net* menggunakan arsitektur *ResNet*-50. Kontribusi penelitian ini adalah mengetahui proses deteksi pengemudi dan sabuk pengaman dengan Model *NADS-Net* menggunakan arsitektur *ResNet*-50.

(Kashevnik, 2020), penelitian ini berjudul *Seat Belt Fastness Detection Based on Image Analysis from Vehicle In-abin Camera*. Tujuan penelitian ini adalah untuk deteksi tahan luntur sabuk pengaman dengan model YOLO. dengan tujuan utamanya adalah untuk membedakan sabuk pengaman diletakkan dengan benar, sabuk pengaman terpasang di belakang, dan sabuk pengaman tidak dipasang sama sekali. Kontribusi penelitian ini adalah menganalisis dan mengetahui hasil proses deteksi menggunakan model YOLO dengan membedakan sabuk pengaman diletakkan dengan benar, sabuk pengaman terpasang di belakang, dan sabuk pengaman tidak dipasang sama sekali.

(Hosam, 2020), penelitian ini berjudul *Deep learning-based car seatbelt classifier resilient to weather conditions.* Tujuan penelitan ini adalah untuk mendeteksi sabuk pengaman dengan akurasi deteksi sangat tinggi, kondisi cuaca yang tepat harus dideteksi sebelum menerapkan jaringan saraf yang dalam model klasifikasi. Pendekatan S-AlexNet yang diusulkan memiliki dua tahap, yaitu pelatihan dan klasifikasi. Dalam proses pelatihan, lebih banyak dari satu classifier CNN akan dilatih, satu classifier akan dilatih untuk kondisi cuaca tunggal. Dalam proses klasifikasi, pertama, sensor akan mendeteksi kondisi cuaca dan kemudian klasifikasi dilakukan dengan menggunakan model CNN terlatih yang sesuai. Pada peneilitian ini berfokus untuk mendeteksi kondisi cuaca pada mobil. Kontribusi penelitian ini adalah menganalisis dan mengetahui proses untuk deteksi kondisi cuaca pada mobil.

(Ale, L., Zhang, N., & Li, L., 2018), Penelitian ini berjudul *Road damage detection using RetinaNet.* Tujuan penelitian ini adalah untuk mendeteksi kerusakan jalan dengan model RetinaNet. Proses yang dilakukan meliputi proses konvolusi, ekstraksi Feature map FPN , proses regresi box dan proses klasifikasi. kontribusi penelitian ini adalah menganalisi dan mengetahui proses untuk deteksi kerusakan pada jalan.

(Eraqi, H. M., Abouelnaga, Y., Saad, M. H., & Moustafa, M. N., 2019), Penelitian ini berjudul *Driver distraction identification with an ensemble of convolutional neural networks.* Tujuan penelitian ini adalah untuk mengidentifikasi aktivitas gangguan saat mengemudi. Penelitian ini menerapkan metode CNN dan SVM untuk mendeteksi aktivitas gangguan saat mengemudi. Kontribusi penelitian ini adalah mengetahui proses untuk mendeteksi aktivitas gangguan saat mengemudi dengan menerapkan metode CNN.

(Sukardi, Z. A., Risaldi, M., & Guna, S. A., 2017), Penelitian ini berjudul Klasifikasi Penentuan Gambar Berbasis *Tensorflow* dan *Framework* dengan Algoritma CNN. Pada penelitian ini melakukan klasifikasi label pada gambar dengan menggunakan model *VGG-Net.* Pada penelitian ini dengan klasifikasi gambar dengan max 2 label didapat hasil akurasi sebesar 93% dan dengan proses klasfikasi melebihi 2 label pada objek gambar dihasilkan akurasi sebesar 63%. Kontribusi penelitian ini

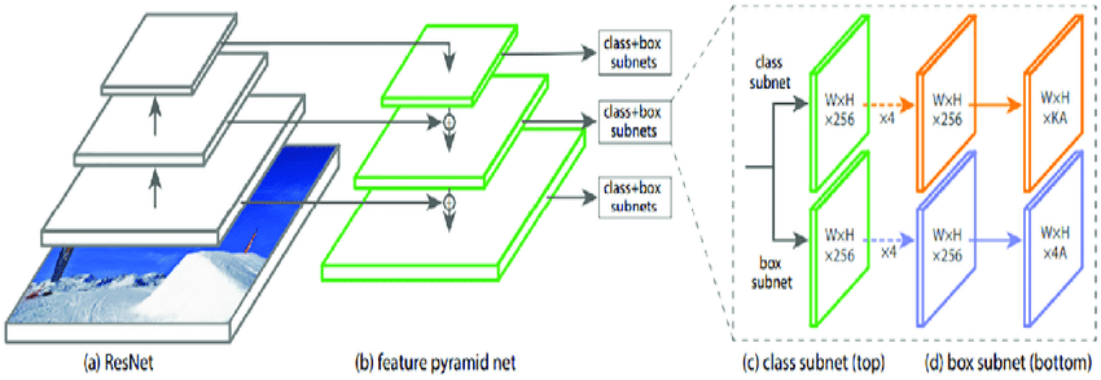
adalah mengetahui proses klasifikasi penentuan gambar dengan menerapkan algoritma CNN menggunakan model *VGG-Net*.

2.2. Deep Learning

Deep learning adalah suatu bidang dari *machine learning* yang diperkenalkan pada tahun 1986, kemudian pada tahun 2000 *deep learning* diterapkan pada metode jaringan syaraf tiruan (Jurgen, 2015). Metode *deep learning* tersusun atas layer yang bertingkat/dalam untuk mempelajari suatu ciri dari data dengan level abstraksi yang bertingkat (LeCun, Y., Bengio, Y., & Hinton, G., 2015). Pendekatan *deep learning* memungkinkan komputer untuk mempelajari suatu model yang rumit dengan cara membangun suatu model yang sederhana (Goodfellow, I., Bengio, Y., & Courville, A., 2016). Berdasarkan hal tersebut, *deep learning* dapat dikatakan sebagai turunan dari *machine learning*, dimana metode ini terdiri dari banyak tingkatan proses informasi non-linear dan abstraksi untuk dapat melakukan *supervised, unsupervised learning* dan representation, klasifikasi, dan pengenalan pola (Deng, L., & Yu, D., 2014).

2.3. RetinaNet

RetinaNet adalah jaringan tunggal dan terpadu yang terdiri dari satu jaringan *backbone* dan dua *subnetwork*. Jaringan *backbone* bertanggung jawab untuk menghitung *feature map* secara konvolusional pada seluruh *citra*. Kemudian pada *subnetwork* pertama bertugas untuk melakukan klasifikasi pada objek hasil konvolusi dari jaringan *backbone*, dan pada *subnetwork* kedua bertugas untuk membentuk *regresi bounding box*. Kedua *subnetwork* memiliki desain yang sederhana yang disiapkan untuk melakukan proses deteksi secara menyeluruh dalam satu tahap. *RetinaNet* mengadopsi FPN sebagai jaringan *backbone* yang digunakannya. Setiap tingkatan pada piramida, dapat digunakan untuk mendeteksi objek dalam skala yang berbeda-beda. FPN meningkatkan prediksi pada skala yang beragam pada *fully connected network* (FCN) (Jung, H., Kim, B., Lee, I., 2018).

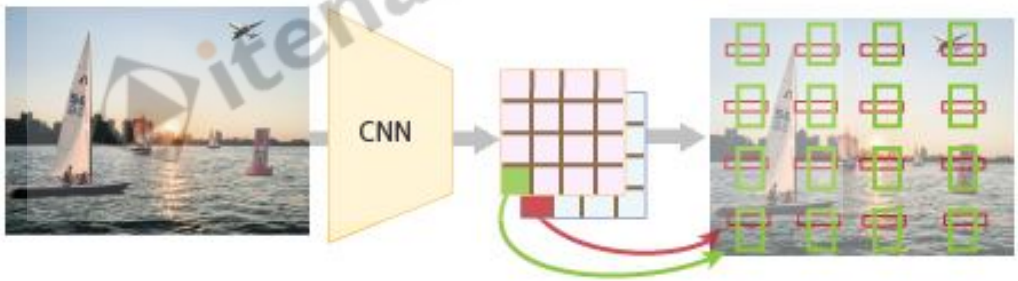


Gambar 2. 1 Arsitektur *RetinaNet*

Sumber : (Jung, H., Kim, B., Lee, I., 2018)

2.3.1. *Anchor*

Pada Gambar 2.2 diilustrasikan prediksi anchor untuk menangkap skala dan rasio pada objek yang akan dideteksi.



Gambar 2. 2 Anchor Box

Sumber : (MathWorks, 2020)

Anchor box adalah satu set kotak pembatas yang telah ditentukan dengan tinggi dan lebar tertentu. *Anchor box* ini didefinisikan untuk menangkap skala dan rasio aspek kelas objek yang akan dideteksi dan biasanya dipilih berdasarkan ukuran objek dalam kumpulan data pelatihan (MathWorks, 2020). Pada bentuk anchor pada jaringan FPN memiliki area berukuran mulai dari 32^2 hingga 512^2 , pada setiap tingkatan pyramid memiliki aspek rasio {1:2, 1:1, 2:1}.

2.3.2. *Classification*

Classification adalah proses memprediksi adanya objek pada setiap posisi spasial setiap *anchor* A dan *class* objek K (Lin, 2017). jaringan subnet klasifikasi ini merupakan jaringan *Fully Connected* kecil yang berfungsi untuk mengklasifikasikan atau mendeteksi objek berdasarkan dataset yang dibuat. Pada proses ini terdapat beberapa tahapan, tahap pertama yaitu mengambil input citra hasil *ekstraksi feature map,* tahap kedua yaitu melakukan proses *konvolusi* 3x3 sebanyak 4 kali, tahap ketiga yaitu melakukan proses *aktivasi ReLU,* tahap keempat melakukan proses *konvolusi* 3x3 dengan menggunakan filter K A per lokasi spasial, tahap terakhir yaitu melakukan proses *aktivasi ReLU*.

2.3.3. Regression

Regresi adalah memprediksi satu variabel atau lebih variabel dengan cara *supervised learning (Wahyu, 2020)*. Pada penelitian ini dimana nilai proses *regresi* akan memprediksi hasil dalam pembentukan *bounding box* dalam mendeteksi objek penggunaan sabuk pengaman pada pengemudi mobil.

2.3.4. *Box Regression*

Box Regression adalah pembentukan *bounding box* pada objek yang terdeteksi dan meregresi setiap kelebihan nilai pada *bounding box* objek yang terdeteksi (MathWorks, 2020). Jaringan subnet *box regression* ini sama seperti jaringan subnet *classification*, tetapi terdapat perbedaan pada langkah terakhir yaitu dilakukan proses operasi *linear* 4A per lokasi *spasial*.

2.3.5. *Focal Loss*

Focal Loss dibuat untuk mengatasi proses deteksi objek dengan satu tahap di mana ada ketidakseimbangan *ekstrem* antara latar depan dan latar belakang selama pelatihan (misalnya., 1: 1000) (Lin, 2017). Untuk mengatasi hal tersebut digunakan fungsi *focal loss* dengan yang dinotasikan pada Persamaan 2.1 sebagai berikut:

$$FL(pt) = -\alpha(1 - pt)^{\gamma} \log(pt) \quad (2.1)$$

Fungsi *focal loss* digunakan untuk menghitung nilai *loss* pada *ground truth* label kelas yang terdeteksi. Dalam melakukan proses *focal loss* menggunakan parameter dengan nilai γ yang digunakan sebesar 2 dan nilai α yang digunakan sebesar 0,25 untuk mendapatkan hasil maksimal pada penggunaan fungsi focal loss. (Lin, 2017).

Berikut ini merupakan ilustrasi penggunaan fungsi *focal loss*. Probabilitas penentuan sebuah sampel terhadap *ground truth* label sebesar 0,9. Maka nilai *loss* yang dihasilkan sebagai berikut:

- $FL = -0{,}25(1 - 0{,}9)2 \log(0{,}9)$

  $= -0{,}25\ x\ 0.01\ x\ (-0{,}04576)$

  $= 0{,}000114$

Nilai loss yang dihasilkan oleh fungsi *focal loss* sebesar 0,000114.

2.4. *Preprocessing*

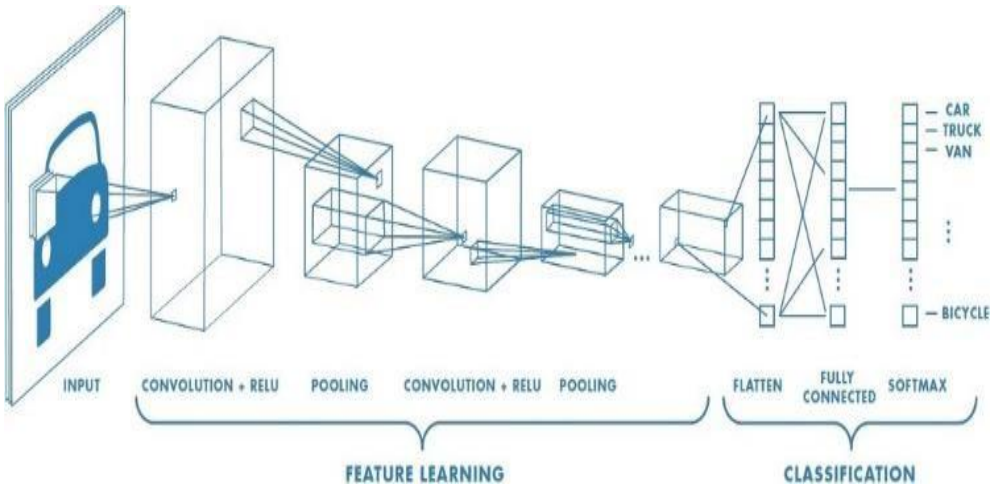
Preprocessing adalah proses ekstraksi matriks citra dengan cara mengurangi *matriks citra BGR* terhadap *filter mode caffe* (Fizyr, 2019), Tujuan *matriks citra* dikurangi *filter mode café* adalah untuk membuat *zero padding* pada setiap warna *channel* warna agar sesuai dengan standar dataset *ImageNet* tanpa proses *scaling*. *Filter mode caffe* dilakukan dengan rumus pada Persamaan 2.2.

B[…,0] - 103,939
G[…,1] - 116,779
R[…,2] - 123,68 (2. 2)

BGR pada Persamaan 2.2 merupakan channel citra

2.5. *Convolutional Neural Network*

Pada Gambar 2.3 dibawah ini merupakan gambaran arsitektur secara umum CNN.



Gambar 2. 3 Proses *Convolutional Neural Network*

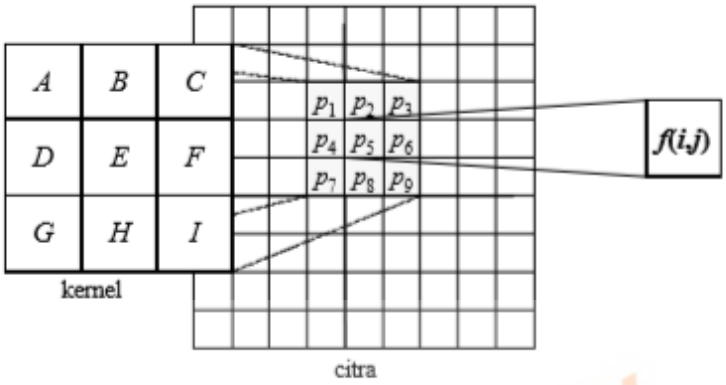
Sumber : (Works, 2019)

Convolutional Neural Network adalah salah satu metode *deep learning* yang paling terkemuka, dimana *multiple layer* dilatih dengan cara yang kuat. Metode ini merupakan metode yang sangat efektif dan biasa digunakan pada aplikasi computer vision (Liu, Y., Guo, Y., Wu, S., & Lew, M. S., 2015). Pada perkembangan CNN terdapat banyak arsitektur yang sering digunakan seperti *AlexNet, VGGNet, GoogLeNet, ResNet*, dll. Secara umum terdapat 3 layer utama pada CNN, yaitu proses *konvolusi*, proses *pooling*, dan proses *fully connected layer*. Pada proses *konvolusi* dilakukan mengalikan nilai masukan citra dengan *filter/kernel* sehingga menghasilkan nilai fitur dari sebuah citra. Kemudian fungsi *pooling layer* adalah untuk memperkecil dimensi dari fitur yang telah didapat dari proses convolutional layer. Sedangkan pada *fully connected layer*, citra yang telah diperkecil pada pooling layer dirubah menjadi 1 dimensi dan diklasifikasikan berdasarkan data latih.

2.5.1. *Convolution Layer*

Dalam pengolahan citra, *konvolusi* yaitu mengalikan nilai masukan citra dengan *filter* (Munir, R., 2004). *Filter* bergerak secara berurutan dari sudut kiri atas sampai ke sudut kanan bawah. Gerakan tersebut bernama *strides*. Jika kernel bergerak satu kolom dari kiri ke arah kanan maka gerakan tersebut disebut 1 *strides*. Jika filter bergerak 2

kolom maka disebut 2 *strides* (Suartika E. P, 2016). Pada Gambar 2.4 diilustrasikan operasi konvolusi.



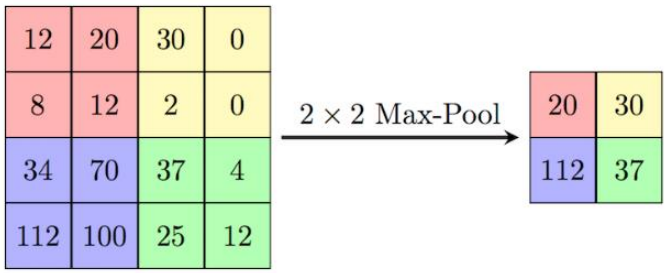
$$f(i,j) = A\,p_1 + B\,p_2 + C\,p_3 + D\,p_4 + E\,p_5 + F\,p_6 + G\,p_7 + H\,p_8 + I\,p_9$$

Gambar 2. 4 Ilustrasi Operasi *Konvolusi*

Sumber : (Munir, R., 2004)

2.5.2. *Pooling*

fungsi *pooling layer* adalah untuk memperkecil dimensi dari fitur yang telah didapat dari proses convolutional layer.



Gambar 2. 5 Proses *Max Pooling*

Sumber : (StackExchange, 2020)

Dalam penerapannya, *pooling layer* terdapat 2 macam fungsi yaitu *max pooling* dan *average pooling*. *Max pooling* adalah proses yang mengambil nilai terbesar pada

setiap pixel citra, sebagai contoh seperti pada Gambar 2.5. *Average Pooling* adalah proses yang mengambil nilai rata-rata setiap *pixel citra* (Nishant, 2018).

2.5.3 *Zero Padding*

Zero padding adalah proses penambahan dimensi pada sisi *matriks citra* dengan angka 0, sehingga dimensi *matriks citra* menjadi lebih besar (Matsui, 2020). Pada Gambar 2.6 diilustrasikan mengenai operasi *zero padding* yang merubah dimensi matriks 8x8 menjadi 10x10.

Gambar 2. 6 Proses *Zero Padding*

Sumber : (Matsui, 2020)

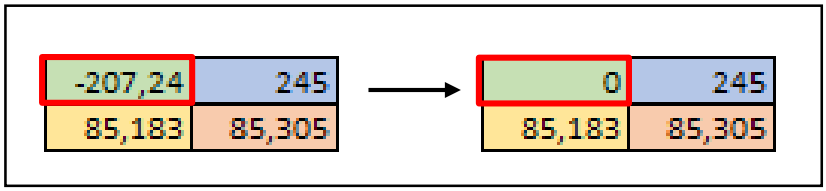
Pada Gambar 2.6 menunjukan penambahan 1 *piksel citra* dengan jumlah *piksel* 0. Tujuan dilakukan proses *zero padding* untuk menambah dimensi *citra* pada setiap *channel* warna *citra* sehingga dimensi *matriks citra* menjadi lebih besar.

2.5.4. *ReLU Activation*

ReLU (Rectified Linear Unit) adalah fungsi linear yang digunakan untuk mengubah nilai negatif menjadi 0. Fungsi *ReLu Activation* pertama kali diperkenalkan oleh Geoffrey Hinton dan Vinod Nair untuk menggantikan fungsi aktivasi *sigmoid* (Pedamonti, 2018). Fungsi *ReLU Activation* dirumuskan pada Persamaan 1.3.

$$ReLu(x) = \max(0, x) \quad (2.3)$$

Pada Gambar 2.7 merupakan proses *ReLU Activation* dengan mengubah nilai negatif menjadi 0.



Gambar 2. 7 Proses ReLU Activation

2.6. Jaringan Arsitektur *Residual Network(ResNet)*

Residual Network adalah jaringan *residual* yang memiliki jaringan yang dalam. Jaringan terdalam dari *ResNet* berjumlah 152 lapisan. Dalam arsitektur *ResNet* memiliki 5 layer, layer pertama yaitu berjumlah 18, layer kedua berjumlah 34, layer ketiga berjumlah 50, layer keempat berjumlah 101, dan layer kelima berjumlah 152. Setiap layer memiliki jumlah kedalaman konvolusi yang berbeda. salah satunya pada *ResNet-152*, proses yang tahap pertama dilakukan *konvolusi* 7x7 dengan jumlah *filter* sebanyak 64 dengan pergeseran perhitungan pixel sebesar 2 *stride*(kolom), tahap kedua dilakukan proses *max pooling* dengan ukuran *matriks* 3x3 dengan pergeseran perhitungan pixel sebesar 2 *stride*, tahap ketiga dilakukan *konvolusi* dengan *matriks* 1x1 dengan jumlah *filter* sebanyak 64, *matriks* 1x1 dengan jumlah *filter* sebanyak 64, *matriks* 3x3 dengan jumlah *filter* sebanyak 64, *matriks* 1x1 dengan jumlah *filter* sebanyak 256, pada tahap ketiga ini dilakukan secara berulang sebanyak 3x. proses ini dilakukan sampai tingkat proses *conv5_x* pada ResNet dengan jumlah layer 152. Sehingga menghasilkan *feature map/bobot* untuk deteksi objek berdasarkan dataset yang dimiliki. Pada Gambar 2.8 merupakan lapisan jaringan pada arsitektur *residual network.*

irenas library

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Gambar 2. 8 Arsitektur dan Lapisan ResNet

Sumber : (Team, 2019)

2.7. Pengujian Kinerja

Pengujian kinerja sistem pendeteksian objek dilakukan dengan mengukur *precision, recall, f1 score* dan akurasi **(Karlita, et al., 2019)**. *Precision* adalah tingkat ketepatan antara informasi dengan hasil yang dideteksi oleh sistem. *Recall* adalah tingkat keberhasilan sistem dalam menemukan jawaban. *F1 Score* adalah perbandingan rata-rata nilai *precision* dan nilai *recall* yang dibobotkan. Akurasi adalah tingkat kesamaan dari hasil nilai prediksi dengan nilai sesungguhnya. *Precision* dirumuskan pada Persamaan 2.4, *recall* dirumuskan pada Persaman 2.5, *F1 Score* dirumuskan pada Persamaan 2.6 dan akurasi dirumuskan pada Persamaan 2.7.

$$Precision = \frac{\text{TP}}{\text{TP+FP}} * 100\% \quad (2.4)$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} * 100\% \quad (2.5)$$

$$F1\ Score = 2 * \frac{(Precision * Recall)}{(Percision + Recall)} * 100\% \quad (2.6)$$

$$Accuracy \quad \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} * 100\% \quad (2.7)$$