BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Pada peneliti ini akan dilakukan eksplorasi literatur sebagai panduan dan rekomendasi untuk menyelami lebih dalam perjalanan penelitian yang akan dilakukan. Referensi ini berupa jurnal-jurnal sebelumnya yang mengulas aspek-aspek terkait dengan isu yang menjadi fokus penelitian dan kerangka kerja yang dipilih. Di bawah ini disajikan beberapa studi sebelumnya yang relevan dengan konteks permasalahan yang akan diteliti oleh peneliti.

Pada penelitian yang berjudul "Deep Learning Application in Dental Caries Detection Using Intraoral Photos Taken by Smartphones" yang bertujuan untuk menerapkan pendekatan *deep learning* dalam mendiagnosis karies gigi melalui gambar smartphone oleh Mai Thi Giang Thanh dkk pada tahun 2022[17]. Pada penelitian tersebut digunakan *Faster R-CNN, YOLOv3, RetinaNet, dan Single-Shot Multi-Box Detector (SSD)*. Data yang digunakan ialah 1902 foto permukaan gigi yang diambil dengan *iPhone* 7 dari 695 orang yang kemudian dilabeli. Dari penelitian tersebut didapatkan karies gigi, *YOLOv3* dan *Faster R-CNN* menunjukkan sensitivitas tertinggi di antara empat model yang diuji, masing-masing sebesar 87,4% dan 71,4%.

Pada penelitian yang berjudul "Development and evaluation of deep learning for screening dental caries from oral photographs" yang bertujuan untuk mengembangkan dan mengevaluasi kinerja sistem *deep learning* berbasis CNN untuk mendeteksi karies gigi dari foto oral oleh Xuan Zhang dkk pada tahun 2020[24]. Metode yang digunakan pada penelitian tersebut adalah *Single Shot MultiBox Detector (SSD)* dan *VGG-1.* Data yang digunakan diperoleh dari Departemen Periodontik, ortodontik dan endodontik, Rumah Sakit Stomatologi Nanjing, Universitas Nanjing. Hasil dari penelitian tersebut menunjukkan *Area Under Certitude* (AUC) klasifikasi sebesar 85,65% (interval kepercayaan 95%: 82,48% hingga 88,71%).

Pada penelitian yang berjudul "Automatic detection of periodontal compromised teeth in digital panoramic radiographs using faster regional convolutional neural networks" yang bertujuan untuk mengusulkan penggunaan metode pendeteksian objek berbasis *deep learning* untuk mengidentifikasi gigi yang terdampak oleh periodontal pada citra radiografi panoramik digital oleh Bhornsawan Thanathornwong dkk pada tahun 2020[18]. Pada penelitian ini algoritma *Faster Regional Convolutional Neural Network (faster R-CNN)* digunakan. Data yang digunakan ialah 100 gambar radiografi panoramik digital anonim yang didapat kan dari basis data registri kedokteran gigi umum institusional penulis dengan persetujuan dari komite etik institusional (DENTSWU-EC16/2561). Model pada penelitian tersebut menghasilkan presisi rata-rata 0,81, sensitivitas 0,84, spesifisitas 0,88 dan *F-measure* 0,81.

Pada penelitian yang berjudul "Detection and diagnosis of dental caries using a deep learning-based convolutional neural network algorithm" yang bertujuan untuk mengevaluasi kemanjuran algoritma deep CNN untuk deteksi dan diagnosis karies gigi pada radiografi periapical oleh Jae-Hong Lee dkk pada tahun 2018[23]. Pada penelitian ini *transfer learning* menggunakan GoogLeNet Inception v3 CNN network digunakan. Dataset yang digunakan ialah 3000 gambar radiografi periapical dari Rumah Sakit Gigi Daejeon, Universitas Wonkwang. Hasil dari penelitian ialah model dengan akurasi 82,0% (75,5–87,1), sensitivitas 81,0% (74,5–86,1), spesifisitas 83,0% (76,5–88,1), PPV 82,7% (76,1 –87.9), dan NPV 81.4% (75.0–86.4) pada gigi Geraham dan Geraham depan.

Pada penelitian yang berjudul "SAR Ship Detection Based On Resnet And Transfer Learning" yang bertujuan untuk membuat model deteksi kapal SAR oleh Yong Li dkk pada tahun 2019[25]. Metode yang digunakan oleh peneliti pada penelitian ini adalah *transfer learning* menggunakan model *pre-trained ResNet50*. Penelitian ini menggunakan data dari satelit *Sentinel-1, RADARSAT-2,* dan *Gaofen-3 sebagai dataset*. Penelitian ini yang metode yang diusulkan oleh penulis diuji pada dataset kapal SAR umum dan mencapai *average precision* sebesar 94,7%.

Pada penelitian yang berjudul "Heartbeats Classification Using Hybrid Time-Frequency Analysis and Transfer Learning Based on ResNet" yang bertujuan untuk mengusulkan metode klasifikasi detak jantung baru menggunakan analisis frekuensi waktu hibrida dan pembelajaran transfer berdasarkan ResNet-101 oleh Yatao Zhang dkk pada tahun 2021[22]. Pada penelitian ini metode yang digunakan ialah *transfer learning* menggunakan model *pre-trained ResNet101*. Dataset yang digunakan pada penelitian ini berasal dari basis data aritmia MIT/BIH di PhysioNet. Hasil pada penelitian ini menunjukkan nilai rata-rata untuk akurasi, *sensitivity, specificity, predictive value* dan *F1-score* pada set tes untuk 14 jenis denyut dari basis data aritmia MIT-BIH adalah 99,75%, 91,36%, 99,85%, 90,81%, dan 0,9016.

Pada penelitian yang berjudul "A Smart Dental Health-IoT Platform Based on Intelligent Hardware, Deep Learning, and Mobile Terminal" yang bertujuan untuk mengeksplorasi kelayakan penerapannya pada perawatan kesehatan gigi di rumah (dalam sudut pandang *Machine Learning* tujuan dari penelitian ini ialah membuat model yang dapat mendeteksi 7 macam penyakit gigi) oleh Lizheng Liu dkk pada tahun 2020[19]. Pada penelitian ini algoritma *MASK R-CNN* digunakan sebagai metode. Data yang digunakan ialah 12.600 ribu gambar klinik yang didapatkan dari klinik gigi (seperti klinik gigi *Beijing Mei-Xiao*). Hasil yang didapatkan model yang dilatih oleh MASK R-CNN dikembangkan untuk mendeteksi dan mengklasifikasikan 7 penyakit gigi yang berbeda termasuk gigi berlubang, plak gigi, uorosis, dan penyakit periodontal, dengan akurasi diagnosis mencapai hingga 90%.

Pada penelitian uang yang berjudul "A deep learning approach to automatic teeth detection and numbering based on object detection in dental periapical flms" yang bertujuan untuk membuat model *Machine Learning* dengan pendekatan *deep learning* untuk deteksi dan penomoran gigi otomatis dalam film periapikal gigi oleh Hu Chen dkk pada tahun 2019[28]. Pada penelitian ini *Faster R-CNN* dan *Dense Neural Network* (DNN) digunakan sebagai metode. Data yang digunakan ialah 1.250 film periapikal gigi digital dikumpulkan dari Sekolah Universitas Peking dan Rumah Sakit Stomatologi. Hasil pada penelitian ini menunjukkan bahwa presisi dan *recall* melebihi 90% dan nilai rata-rata *intersection-over-union* (IOU) antara

kotak yang terdeteksi dan kebenaran di lapangan mencapai 91%. Hasil dari penelitian menunjukkan bahwa model sudah bekerja mendekati tingkat dokter gigi junior.

Pada penelitian yang yang berjudul "Transfer Learning with ResNet-50 for Malaria Cell-Image Classification" yang bertujuan untuk menyajikan pendekatan klasifikasi citra cell Malaria menggunakan *transfer learning* berdasarkan arsitektur *ResNet-50* oleh A. Sai Bharadwaj Reddy dkk pada tahun 2019[29]. Pada penelitian metode yang digunakan adalah *transfer learning* menggunakan *ResNet50*. Pada penelitian ini data yang digunakan adalah citra cell yang sudah terinfeksi dan belum terinfeksi malaria dari *National Library of Medicine (NLM)*. Hasil yang didapatkan dari hasil penelitian ini adalah akurasi *training*, dan validasi adalah 95.91%, dan 95.4%.

Pada penelitian yang yang berjudul "DeNTNet: Deep Neural Transfer Network for the detection of periodontal bone loss using panoramic dental radiographs" yang bertujuan merancang model berbasis *deep learning* untuk mengembangkan sistem pendukung diagnostik otomatis yang mendeteksi kehilangan tulang periodontal pada radiografi gigi panoramik oleh Jaeyoung Kim dkk pada 2019[30]. Pada penelitian ini deep *convolutional neural network (CNNs)* diterapkan menggunakan *transfer learning* dan pengetahuan awal klinis untuk mengatasi variasi morfologi lesi dan kumpulan data pelatihan yang tidak seimbang. Data yang digunakan merupakan 12.179 gambar radiografi gigi panoramik dari Rumah Sakit Universitas Anam Korea setelah menghapus informasi pasien yang dapat diidentifikasi selama periode antara 1 Januari 2014 dan 14 Februari 2016. Hasil yang didapatkan adalah jika dibandingkan dengan dokter gigi, *DeNTNet* mencapai skor F1 0,75 pada set tes, sedangkan kinerja rata-rata dokter gigi adalah 0,69.

Pada penelitian yang yang berjudul "Evaluation of Transfer Learning with Deep Convolutional Neural Networks for Screening Osteoporosis in Dental Panoramic Radiographs" yang bertujuan untuk mengevaluasi kinerja diskriminatif *Deep Convolutional Neural Networks* (CNN), yang digunakan dengan berbagai strategi *transfer learning*, pada klasifikasi fitur spesifik osteoporosis di *Dental*

panoramic radiographs (DPR) oleh Ki-Sun Lee, dkk pada tahun 2020[31]. Metode yang digunakan pada penelitian ini adalah VGG-16, model VGG-16 (VGG-16_TF), dan penyempurnaan dengan model pembelajaran transfer (VGG-16_TF_FT). Dataset yang digunakan berisi 680 gambar dari pasien menjalani pemeriksaan kepadatan mineral tulang rangka dan pemeriksaan radiografi panoramik digital di Rumah Sakit Ansan Universitas Korea antara tahun 2009 dan 2018. Hasil dari penelitian menunjukkan model dengan kinerja terbaik mencapai area keseluruhan di bawah karakteristik pengoperasian penerima sebesar 0,858 dan dalam studi ini, *transfer learning* dan *fine-tuning* meningkatkan kinerja *Deep CNN* untuk skrining osteoporosis pada citra DPRs. Hasil ini menunjukkan bahwa penilaian citra DPR berbasis *deep learning* dapat berguna dan dapat diandalkan dalam skrining otomatis pasien osteoporosis.

Pada Penelitian yang yang berjudul "Efficacy of deep convolutional neural network algorithm for the identification and classification of dental implant systems, using panoramic and periapical radiographs" yang bertujuan untuk mengevaluasi kemanjuran algoritma *deep CNN* untuk identifikasi dan klasifikasi sistem implan gigi oleh Jae-Hong Lee, dkk pada tahun 2020[32]. Pada penelitian ini pra-pemrosesan gambar dan teknik transfer learning menggunakan arsitektur *Deep CNN* yang telah *fine-tuned* dan dilatih sebelumnya menggunakan GoogLeNet Inception-v3. Dataset yang digunakan adalah gambar radiografi panoramik dan periapikal mentah (INFINITT PACS, Infinitt, Seoul, Korea) dari pasien yang menjalani perawatan implan gigi di rumah sakit gigi diperoleh antara Januari 2010 dan Desember 2019. Hasil dari penelitian ini menemukan bahwa arsitektur *Deep CNN* (AUC=0,971, interval kepercayaan 95% 0,963–0,978) sedangkan periodontitis bersertifikat (AUC=0,925, interval kepercayaan 95% 0,913–0,935) menunjukkan akurasi model klasifikasi dapat diandalkan.

Tabel 2. 1 Penelitian Terdahulu

No	Judul	Penulis	Metode	Hasil
1.	Deep Learning Application in Dental Caries Detection Using Intraoral Photos Taken by Smartphones	Mai Thi Giang Thanh, dkk	*Faster R-CNN, YOLOv3, RetinaNet, SSD*	Untuk karies gigi, YOLOv3 dan Faster R-CNN menunjukkan sensitivitas tertinggi di antara empat model yang diuji, masing-masing sebesar 87,4% dan 71,4%.
2.	Development and evaluation of deep learning for screening dental caries from oral photographs	Xuan Zhang, dkk.	*Single Shot MultiBox Detector (SSD), VGG-16*	Sistem menunjukkan area klasifikasi di bawah kurva (AUC) sebesar 85,65% (interval kepercayaan 95%: 82,48% hingga 88,71%).
3.	Automatic detection of periodontal compromised teeth in digital panoramic radiographs using faster regional convolutional neural networks	Bhornsawan Thanathornwong, dkk	*Faster Regional Convolutional Neural Network (faster R-CNN)*	Model pada penelitian tersebut menghasilkan presisi rata-rata 0,81, sensitivitas 0,84, spesifisitas 0,88 dan F-measure 0,81.
4.	Detection and diagnosis of dental caries using a deep	Jae-Hong Lee, dkk	*GoogLeNet Inception-v3*	Model memiliki akurasi 82,0% (75,5–87,1), sensitivitasnya 81,0% (74,5–86,1), spesifisitasnya

No	Judul	Penulis	Metode	Hasil
	learning-based convolutional neural network algorithm			83,0% (76,5–88,1), PPV adalah 82,7% (76,1 – 87.9), dan NPV adalah 81.4% (75.0–86.4) pada gigi Geraham dan Geraham depan.
5.	Sar Ship Detection Based On Resnet And Transfer Learning	Yong Li, dkk	*ResNet50*	Metode yang diusulkan diuji pada dataset kapal SAR umum dan mencapai *average precision* 94,7%.
6.	Heartbeats Classification Using Hybrid Time-Frequency Analysis and Transfer Learning Based on ResNet	Yatao Zhang, dkk	*Resnet101*	Hasil penelitian ini menunjukkan nilai rata-rata untuk akurasi, *sensitivity, specificity, predictive value* dan *F1-score* pada set tes untuk 14 jenis denyut dari basis data aritmia MIT-BIH adalah 99,75%, 91,36%, 99,85%, 90,81%, dan 0,9016.
7.	A Smart Dental Health-IoT Platform Based on Intelligent Hardware, Deep Learning, and Mobile Terminal	Lizheng Liu, dkk	*Mask R-CNN*	Hasil yang didapatkan model yang dilatih oleh MASK R-CNN dikembangkan untuk mendeteksi dan mengklasifikasikan 7 penyakit gigi yang berbeda termasuk gigi berlubang, plak gigi, uorosis, dan penyakit.

No	Judul	Penulis	Metode	Hasil
8.	A deep learning approach to automatic teeth detection and numbering based on object detection in dental periapical films.	Hu Chen, dkk	*Faster R-CNN dan Dense Neural Network (DNN)*	Hasil pada penelitian ini menunjukkan bahwa presisi dan *recall* melebihi 90% dan nilai rata-rata *intersection-over-union* (IOU) antara kotak yang terdeteksi dan kebenaran di lapangan mencapai 91%.
9.	Transfer Learning with ResNet-50 for Malaria Cell-Image Classification	A. Sai Bharadwaj Reddy, dkk	*ResNet50*	Hasil yang didapatkan dari model klasifikasi ialah akurasi pelatihan 95,91%, dan akurasi validasi 95,4%.
10.	DeNTNet: Deep Neural Transfer Network for the detection of periodontal bone loss using panoramic dental radiographs	Jaeyoung Kim, dkk	*Deep Convolutional Neural Network (CNNs)*	Hasil yang didapatkan adalah jika dibandingkan dengan dokter gigi, DeNTNet mencapai skor F1 0,75 pada set tes, sedangkan kinerja rata-rata dokter gigi adalah 0,69.
11.	Evaluation of Transfer Learning with Deep Convolutional Neural Networks for Screening	Ki-Sun Lee, dkk	*CNN3, VGG-16, VGG-16 (VGG-16_TF), dan (VGG-16_TF_FT).*	Hasil dari penelitian menunjukkan model dengan kinerja terbaik mencapai area keseluruhan di bawah karakteristik pengoperasian penerima sebesar 0,858 dan dalam studi ini, *transfer learning* dan *fine-tuning* meningkatkan kinerja

No	Judul	Penulis	Metode	Hasil
	Osteoporosis in Dental Panoramic Radiographs			*Deep CNN* untuk skrining osteoporosis pada citra DPRs.
12.	Efficacy of deep convolutional neural network algorithm for the identification and classification of dental implant systems, using panoramic and periapical radiographs	Jae-Hong Lee, dkk	*GoogLeNet Inception-v3*	Hasil dari penelitian ini menemukan bahwa arsitektur *Deep CNN* (AUC=0,971, interval kepercayaan 95% 0,963–0,978) sedangkan periodontitis bersertifikat (AUC=0,925, interval kepercayaan 95% 0,913–0,935) menunjukkan akurasi model klasifikasi dapat diandalkan.

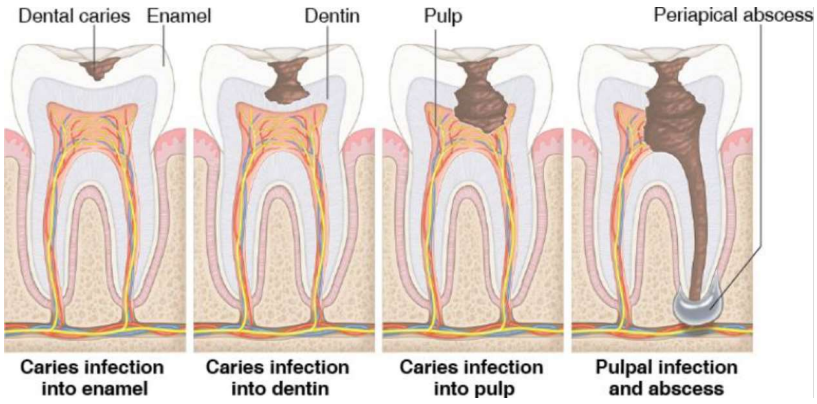
Berdasarkan penelitian terdahulu, *transfer learning* menggunakan model *pre-trained ResNet* dapat menghasilkan akurasi yang baik, mempercepat proses pelatihan, dan meningkatkan performa. Terdapat penelitian terdahulu yang membahas deteksi karies gigi dengan menggunakan berbagai metode di antaranya yaitu *ResNet50*, *Faster R-CNN, YOLOv3, RetinaNet, SSD, Inception-Resnet-v2*[17], *VGG-16*[24], *GoogLeNet Inception-v3*[23]. Terdapat beberapa penelitian dalam berbagai kasus menggunakan *ResNet-50 dan ResNet-101* di berbagai bidang, seperti deteksi kapal[25], klasifikasi detak jantung[22] dan klasifiaksi cell malaria[29], dan banyak lainnya. Implementasi *transfer learning ResNet* untuk objek deteksi pada dataset *COCO* menunjukkan peningkatan sebanyak 6% pada *COCO*

standard metric menunjukkan implementasi ResNet dapat digunakan untuk deteksi objek[33]. Oleh karena itu, pada penelitian ini akan mengimplementasikan *transfer learning Faster R-CNN* dengan *feature extractor ResNet-50* dan *ResNet-101* untuk deteksi karies gigi.

2.2. Dasar Teori

2.2.1. Karies Gigi

Karies gigi adalah penyakit gigi dengan paling umum di dunia. Menurut sebuah studi, karies yang tidak diobati pada gigi permanen merupakan kondisi kesehatan yang paling umum secara global, mempengaruhi 34,1% populasi dunia[34]. Selain itu karies gigi juga mempengaruhi 28.8% populasi yang berusia lima tahun ke atas Di Indonesia berdasarkan *Oral Health Country Profile* untuk Indonesia yang dipublikasikan oleh *World Health Organization*[35]. Karies gigi tanpa perawatan yang tepat dapat menyebabkan penyakit pulpitis dan periapikal[36]. Karies gigi juga merupakan penyebab utama dari rasa nyeri di mulut dan kehilangan gigi[37]. Perkembangan karies dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Diagram perkembangan karies gigi[38].

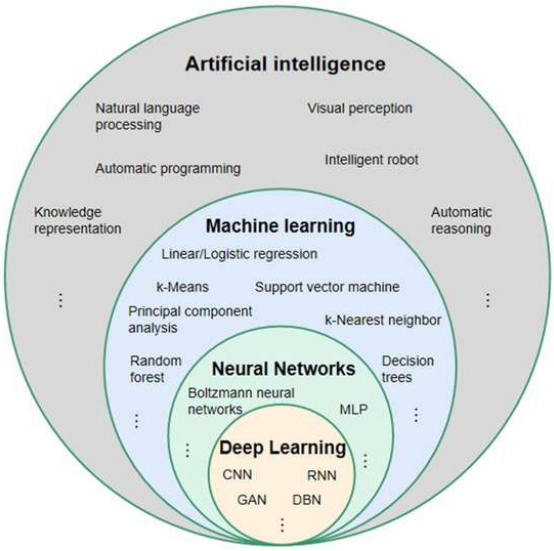
Karies gigi atau *dental caries* adalah kondisi di mana terjadi kerusakan pada jaringan keras gigi akibat produk sampingan berbentuk asam dari proses fermentasi bakteri, yang dihasilkan dari karbohidrat dalam makanan [39]. Karies gigi dimulai dengan pergeseran mikrobiologi di dalam biofilm bakteri (plak gigi) yang menutupi permukaan gigi dan dipengaruhi oleh faktor utama seperti aliran air liur, paparan fluoride, konsumsi gula makanan, dan perilaku pencegahan (membersihkan gigi). Karies gigi pada awalnya bersifat reversibel dan dapat dihentikan pada tahap apa pun, bahkan ketika beberapa dentin atau email hancur (kavitasi), asalkan cukup banyak biofilamen yang dapat dihilangkan[40].

Selain faktor utama ada beberapa faktor-faktor lain yang dapat menyebabkan karies gigi diantarnya adalah faktor individu, keluarga, dan masyarakat yang

meliputi lingkungan sosial dan fisik, perilaku dan layanan kesehatan serta determinan komersial, semuanya bekerja dalam jangka waktu seumur hidup[41]. Lebih detailnya lagi faktor-faktor tersebut meliputi umur, lingkungan fisik dan sosial, penggunaan *fluoride*, kekurangan vitamin, lokasi tempat tinggal, kehamilan, jenis kelamin, diet makanan, perilaku, pendidikan, kebersihan mulut yang buruk, kunjungan ke dokter gigi, perilaku merokok dan banyak lainnya[42][43].

2.2.2. Deep Learning

Deep Learning merupakan bagian dari keluarga besar metodologi *Machine Learning* yang dasarnya berasal dari *artificial neural networks* dan *representation learning*. *Deep learning* adalah kumpulan *classifier* yang bekerja bersama, yang didasarkan pada regresi linier yang diikuti oleh beberapa fungsi aktivasi. Dasarnya sama dengan pendekatan regresi linier statistik tradisional. Satu-satunya perbedaan adalah bahwa ada banyak simpul saraf dalam *deep learning*, bukan hanya satu simpul yang disebut regresi linier dalam pembelajaran statistik tradisional. *Node neural* ini juga dikenal sebagai jaringan neural, dan satu *node clasifier* dikenal sebagai unit neural atau persepsi. Hal lain yang perlu diperhatikan adalah dalam *deep learning* terdapat banyak lapisan antara input dan output. Sebuah lapisan dapat memiliki ratusan atau bahkan ribuan unit saraf[44].



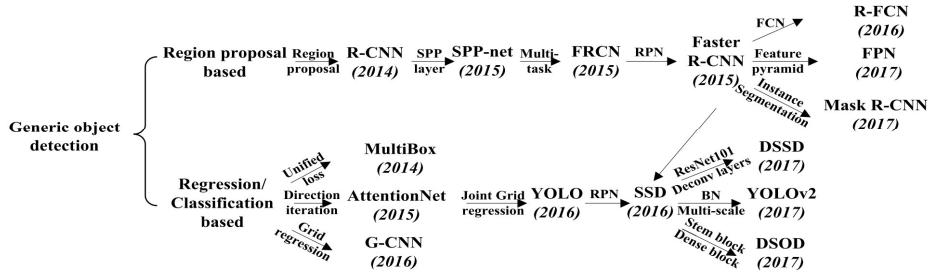
Gambar 2. 2 Deep learning sebagai bagian dari keluarga besar artificial intelligence dan Machine Learning[45].

Lapisan yang berada di antara input dan output dikenal sebagai lapisan tersembunyi dan simpul-simpulnya dikenal sebagai simpul tersembunyi. Kekurangan dari pengklasifikasi pembelajaran mesin tradisional adalah bahwa kita perlu menulis hipotesis yang kompleks sendiri, sementara dalam jaringan saraf dalam, hipotesis tersebut dihasilkan oleh jaringan itu sendiri, yang membuatnya menjadi alat yang ampuh untuk mempelajari hubungan nonlinier secara efektif[44].

2.2.3. Object Detection

Object Detection adalah tugas *computer vision* yang berhubungan dengan pendeteksian *instance* objek visual dari sebuah kelas tertentu (seperti manusia, hewan, atau mobil) dalam citra digital. Tujuan dari *object detection* adalah untuk mengembangkan model dan teknik komputasi yang menjawab pertanyaan: "Objek apa yang ada di mana?". Dua metrik yang paling penting untuk object detection adalah akurasi (akurasi klasifikasi dan akurasi lokalisasi) dan kecepatan[10]. *Object detection* generik bertujuan untuk menemukan dan mengklasifikasikan objek yang ada dalam satu citra dan melabelinya dengan *bounding box* persegi panjang untuk menunjukkan keyakinan keberadaan objek tersebut. *Framework* metode *object detection* generik pada umumnya dapat dikategorikan menjadi dua jenis[13]:

1. *Region proposal based*: *region proposal based* mengikuti alur pendeteksian objek tradisional, yang di mana proposal wilayah terlebih dahulu dihasilkan dan kemudian mengklasifikasikan setiap proposal ke dalam kategori objek yang berbeda. Metode *region proposal based* terutama mencakup *R-CNN, Fast R-CNN, Faster R-CNN, R-FCN, feature pyramid networks* (*FPN*), dll.
2. Regression/classification based: *framework* ini menganggap deteksi objek sebagai masalah regresi atau klasifikasi, mengadopsi *framework* terpadu untuk mencapai hasil akhir (kategori dan lokasi) secara langsung. Metode *regression/classification based* terutama meliputi *YOLO, Single Shot MultiBox Detector (SSD), YOLOv2, AttentionNet, MultiBox*, dll.



Gambar 2. 3 Dua jenis framework object detection: region proposal based and regression/classification based[13].

2.2.4. Convolutional Neural Network (CNN)

Convolutional Neural Networks (*CNNs*) CNN adalah sejenis *feedforward neural network* yang mampu mengekstrak fitur dari data dengan struktur konvolusi[46]. CNN mirip dengan *Artificial Neural Networks (ANN)* tradisional karena terdiri dari neuron yang mengoptimalkan dirinya melalui pembelajaran. Setiap neuron menerima input dan melakukan operasi (seperti produk skalar yang diikuti oleh fungsi non-linear) untuk mengekstrak fitur dari yang diinputkan. Fitur-fitur ini kemudian diteruskan melalui jaringan, dengan output akhir berupa skor kelas. Lapisan terakhir dari jaringan berisi fungsi *loss* yang berhubungan dengan kelas, dan semua tips dan trik yang biasa dikembangkan untuk ANN tradisional masih berlaku[47]. Arsitektur CNN terinspirasi oleh persepsi visual, dengan neuron buatan yang sesuai dengan neuron biologis, kernel CNN mewakili reseptor berbeda yang dapat merespons berbagai fitur, dan fungsi aktivasi yang menyimulasikan fungsi bahwa hanya sinyal listrik saraf yang melebihi ambang batas tertentu yang dapat ditransmisikan ke neuron berikutnya. Tidak seperti metode ekstraksi fitur tradisional, CNN tidak memerlukan ekstraksi fitur manual[48].

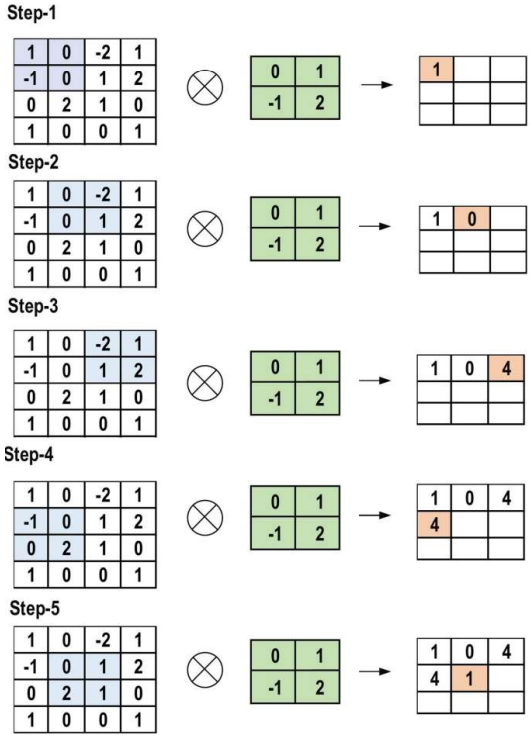
Arsitektur *Convolutional Neural Network* (*CNNs*) sendiri terdiri dari beberapa komponen utama, yaitu:

a. *Convolutional Layer*

   Bagian paling penting dari arsitektur CNN adalah *Convolutional Layer*. *Convolutional Layer* terdiri dari kumpulan filter konvolusi (yang disebut kernel). Proses konvolusi menggunakan filter ini untuk mengekstraksi fitur-fitur penting dari citra input. Filter konvolusi ini

digunakan untuk mengidentifikasi pola-pola tertentu dalam citra input seperti tepi, sudut, atau ciri-ciri lainnya yang relevan. Setelah proses konvolusi, peta fitur keluaran yang dihasilkan akan digunakan sebagai input untuk lapisan-lapisan berikutnya dalam arsitektur CNN untuk mengidentifikasi fitur-fitur yang lebih abstrak dan mengekstrak informasi yang lebih kompleks[49]. Pada *Convolutional Layer*, filter bergeser di atas layer untuk data input yang diberikan. Penjumlahan dari perkalian elemen per elemen dari filter dan bidang reseptif dari input kemudian dihitung sebagai output dari lapisan ini. Kemudian Hasil penjumlahan terbobot ditempatkan sebagai elemen dari lapisan berikutnya[50]. Ada tiga keuntungan utama dari operasi konvolusi, yaitu[51]:

1. Mekanisme pembagian bobot dalam peta fitur yang sama mengurangi jumlah parameter.
2. Konektivitas lokal mempelajari korelasi di antara piksel tetangga.
3. Invarian terhadap lokasi objek.

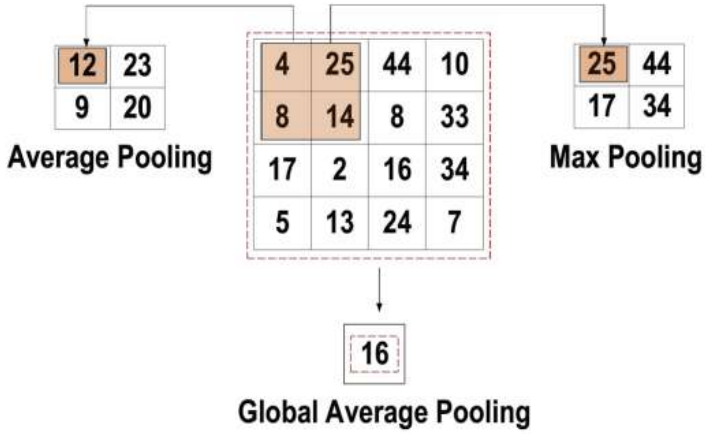


Gambar 2. 4 Proses konvolusi pada Convolutional Layer[49].

Dalam gambar 2.4, table warna hijau muda merepresentasikan kernel 2 × 2, sementara warna biru muda merepresentasikan area yang sama dari gambar input. Kedua warna tersebut dikalikan dan hasilnya ditambahkan, yang diwakili oleh warna oranye terang hingga tidak ada pergeseran lebih lanjut yang memungkinkan. Ketika sudah tabel kanan kemudian dijadikan sebagai nilai entri pada peta fitur keluaran.

b. *Pooling Layer*

Pooling layer digunakan untuk *sub-sampling* atau mengurangi dimensi peta fitur dan parameter jaringan, p*ooling layer* umumnya mengikuti convolutional layer[50]. Mirip dengan *convolutional layer*, *pooling layer* juga dapat mendeteksi fitur yang sama terlepas dari lokasinya dalam citra, karena perhitungannya memperhitungkan piksel tetangga[52]. Secara bersamaan, *pooling layer* mempertahankan mayoritas informasi dominan (atau fitur) dalam setiap langkah tahap *pooling* nya[49]. Ada beberapa jenis *pooling* pada *pooling layer, di mana Average pooling* dan *max pooling* adalah strategi yang paling umum digunakan.



Gambar 2. 5 Proses konvolusi pada Convolutional Layer[49].

Seperti yang ditunjukan pada Gambar 2.5, pada jenis *pooling "average pooling"* nilai yang diperoleh ialah nilai rata-rata dari matriks yang menjalani operasi *pooling*. Sedangkan *max pooling* nilai yang

diperoleh adalah nilai tertinggi dari matriks yang menjalani operasi *pooling*.

c. *Activation Function (non-linearity)*

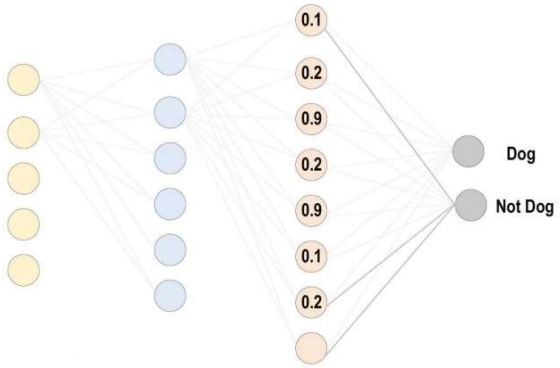
Activation Function adalah fungsi yang digunakan dalam *neural network* untuk menghitung hasil dari penjumlahan bobot input dan bias. Fungsi ini digunakan untuk menentukan apakah sebuah neuron harus diaktifkan atau tidak, atau dengan kata lain, apakah neuron tersebut harus mengirimkan sinyal keluar atau tidak[53]. Fungsi ini mengolah data yang diberikan melalui beberapa proses gradient, biasanya menggunakan metode gradient descent, dan menghasilkan output untuk jaringan saraf. Output tersebut berisi parameter dalam data yang akan digunakan untuk proses selanjutnya. [54]. *Activation Function* dapat berupa linear ataupun non-linear. Jenis *activation Function* berikut ini paling sering digunakan pada arsitektur CNN dan *deep neural networks* lainnya[49]:

1. *Sigmoid*: Input dari *activation Function* ini adalah bilangan real, sedangkan output dibatasi antara nol dan satu. Kurva fungsi *sigmoid* berbentuk S
2. *ReLU*: *ReLU* adalah fungsi yang paling umum digunakan dalam konteks CNN. Fungsi ini mengubah semua nilai input menjadi angka positif. Keuntungan utama dari ReLU adalah beban komputasi yang lebih rendah dibandingkan dengan fungsi aktivasi lainnya.
3. *Leaky ReLU*: Alih-alih menurunkan skala input negatif *ReLU*, fungsi aktivasi ini memastikan input ini tidak pernah diabaikan. Fungsi ini digunakan untuk memecahkan masalah *Dying ReLU*.
4. *Tanh*: Fungsi ini mirip dengan fungsi *sigmoid*, karena inputnya berupa bilangan real, tetapi outputnya dibatasi antara -1 dan 1

d. *Fully Connected Layer*

Umumnya, *layer fully connected* atau *dense* terletak di akhir setiap arsitektur CNN. *Layer* ini biasanya terletak di akhir arsitektur CNN karena digunakan sebagai pengklasifikasi CNN. Di dalam lapisan ini, setiap neuron terhubung dengan semua neuron pada lapisan sebelumnya,

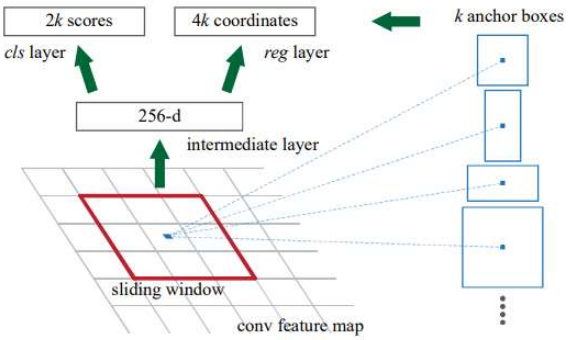
dan pendekatan ini disebut dengan *Fully Connected (FC)*. Layer ini juga mengikuti metode dasar dari jaringan saraf *perceptron multi-layer* konvensional, karena ini adalah jenis ANN umpan-maju[55]. Masukan dari lapisan *Fully Connected (FC)* berasal dari lapisan *pooling* atau lapisan konvolusi terakhir. Masukan ini berbentuk vektor, yang dihasilkan dari peta fitur setelah dilakukan proses *flattening*. Keluaran dari lapisan FC merupakan keluaran akhir CNN[55].



Gambar 2. 6 Fully Connected Layer[49].

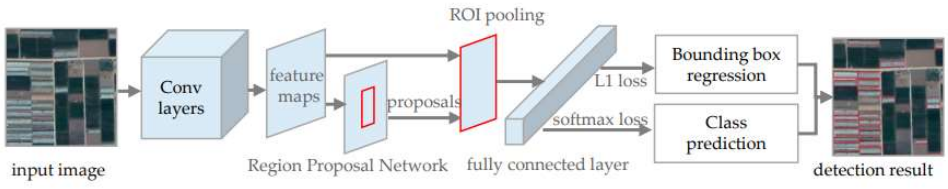
2.2.5. Faster R-CNN

Faster Region-based Convolutional Neural Network (Faster R-CNN) adalah sistem objek deteksi yang terdiri dari dua modul. Modul yang pertama adalah *deep fully convolutional network* atau *Region Proposal Network* (RPN), modul ini bertugas memberikan region proposal. Modul kedua adalah detektor yang melakukan objek deteksi pada region proposal tersebut[16].



Gambar 2. 7 Region Proposal Network [16].

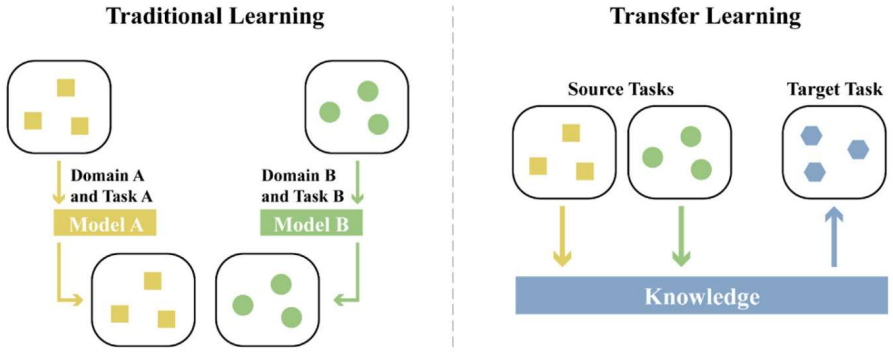
Faster R-CNN bekerja sebagai berikut, pertama-tama *Faster R-CNN* menerima citra sebagai input yang kemudian akan digunakan untuk menghasilkan fitur dengan cara *feature mapping* menggunakan lapisan konvolusi. Fitur ini kemudian akan di bagi pada modul RPN dan detektor. Fitur luaran dari lapisan konvolusi kemudian dikirim ke RPN yang digunakan untuk menghasilkan region proposal[14]. RPN merupakan komponen utama dari *Faster R-CNN* yang di mana bertanggung jawab untuk menghasilkan region proposal dengan menganalisis peta fitur dari konvolusi dan memprediksi skor objektivitas dan kandidat *bounding box*. RPN berfungsi dengan menggeser jendela n x n di atas peta fitur untuk menghasilkan region proposal. Di mana untuk setiap jendela pada peta fitur dapat membuat 9 *anchor* dengan 3 skala dan 3 aspek rasio yang berbeda, kemudian anchor ini dinilai positif atau negatif berdasarkan ketersediaan target/objek. *Anchor* positif dan negatif dipilih secara acak dengan rasio 1:1 sebagai *minibatch* untuk mencegah terjadinya bias, yang digunakan untuk menghasilkan region proposal dengan membandingkannya dengan *ground truth* dari objek pada proses pelatihan[15]. Ilustrasi RPN dapat dilihat pada Gambar 2.7. Pada modul detektor, *ROI pooling layer* memetakan fitur konvolusional yang diekstrak dari region proposal ke dalam vektor fitur berukuran seragam, yang selanjutnya berfungsi sebagai input dari *fully connected layer*. Akhirnya, *softmax* dapat digunakan untuk menentukan sekumpulan nilai probabilitas untuk menentukan apa kategori objek tersebut, sementara regressor digunakan untuk mencari nilai koordinat yang lebih akurat dari kotak pembatas region proposal[15]. Ilustrasi Arsitektur *Faster R-CNN* dapat dilihat pada Gambar 2.8.



Gambar 2. 8 Ilustrasi Arsitektur Faster R-CNN[15].

2.2.6. Transfer Learning

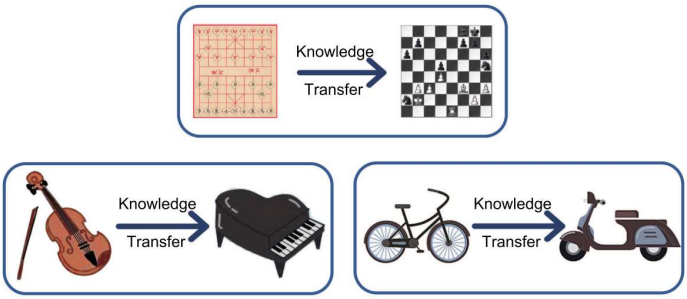
Transfer Learning adalah sebuah teknik *machine learning* di mana perpindahan/*transfer* pengetahuan dari sebuah domain sumber ke domain target[21]. Inti *transfer learning* adalah untuk meningkatkan kemampuan pembelajaran pada tugas target dengan menggunakan pengetahuan yang diperoleh dari tugas sumber atau mentransfer pengetahuan dari domain sumber ke domain target[56]. Gambar 2.9 menggambarkan perbedaan antara pembelajaran tradisional dan pembelajaran transfer. Bagian (a) dari gambar menunjukkan proses kerja pembelajaran tradisional, di mana tugas yang berbeda membutuhkan domain yang berbeda untuk melatih model. Bagian (b) menunjukkan bagaimana pembelajaran transfer dapat menyintesis domain yang berbeda secara keseluruhan, sehingga memungkinkan model untuk memperoleh pengetahuan dari domain sumber dan meningkatkan kinerjanya pada tugas target[57].



Gambar 2. 9 Perbedaan (a) traditional learning dan (b) transfer learning[57].

Seorang manusia sebagai pembelajar memiliki cara yang alami untuk mentransfer pengetahuan di antara tugas-tugas. Di mana manusia mengenali dan menerapkan pengetahuan yang relevan dari pengalaman belajar sebelumnya ketika menghadapi tugas baru. Contohnya dapat dilihat pada Gambar 2.10. di mana seseorang yang telah belajar violin dapat belajar piano lebih cepat daripada yang lain, karena violin dan piano adalah alat musik dan dapat berbagi pengetahuan dasar yang sama[58]. Dengan contoh implementasi nyata sebagai berikut katakanlah sebuah model jaringan saraf tiruan dilatih untuk mengenali berbagai jenis hewan dengan menggunakan dataset besar yang berisi gambar berbagai hewan. Model ini

telah belajar mengekstrak fitur-fitur seperti bentuk, tekstur, dan pola yang umum di berbagai hewan. Sekarang, sebuah model untuk mengidentifikasi jenis anjing tertentu ingin dibuat, tetapi dataset yang dimiliki sedikit. Daripada melatih model baru dari awal dengan data yang terbatas ini, *transfer learning* bisa diterapkan. Dengan menggunakan model pengenalan berbagai jenis hewan yang telah dilatih sebelumnya dan menyempurnakannya pada dataset jenis anjing ini. Lapisan awal model, yang telah mempelajari fitur-fitur umum, dapat dipertahankan, dan hanya lapisan berikutnya yang disesuaikan untuk mengkhususkan diri dalam mengenali jenis anjing.



Gambar 2. 10 Contoh Intuitif transfer learning[58].

Pendekatan terhadap *transfer learning* dapat dikategorikan ke dalam empat kelompok yaitu pendekatan[58]: *feature-based, relational-based, parameter-based, dan instance-based*. Pendekatan *feature-based* mengubah fitur asli untuk membuat representasi fitur baru; pendekatan ini dapat dibagi menjadi dua sub kategori, yaitu *transfer learning* berbasis fitur asimetris dan simetris. Pendekatan asimetris mengubah fitur sumber agar sesuai dengan fitur target. Sebaliknya, pendekatan simetris berusaha menemukan ruang fitur laten yang sama dan kemudian mentransformasikan fitur sumber dan target menjadi representasi fitur baru. Pendekatan *relational-based* berfokus pada masalah dalam domain relasional. Pendekatan tersebut mentransfer hubungan logis atau aturan yang dipelajari dalam domain sumber ke domain target. Pendekatan *parameter-based* mentransfer pengetahuan pada tingkat model/parameter. Pendekatan transfer learning *instance-based* didasarkan pada strategi pembobotan *instance*. Pendekatan *instance-based* didasarkan pada strategi pembobotan instance.

Ada beberapa *pre-trained* model yang secara luas digunakan luas dalam penerapan *transfer learning* di bidang medis. Model tersebut di antaranya adalah *ResNet50, Inception-v3, DensNet, Xception* dan *InceptionResNetV2*[59][60][61]. Model-model tersebut di latih menggunakan *ImageNet*. Dan pada dalam penelitian ini, *ResNet50* dan *ResNet101* akan digunakan sebagai *pre-trained* model.

2.2.7. Deep Residual Learning (Residual Network/ResNet)

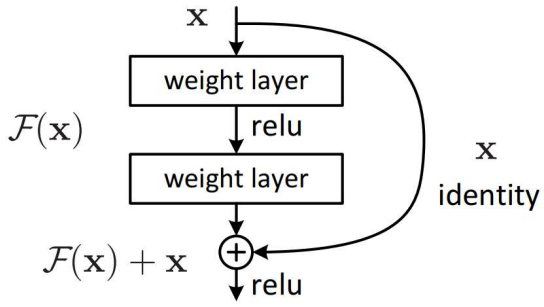
Residual Network atau lebih kerap dipanggil *ResNet* pertama kali diperkenalkan pada tahun 2015 oleh He et al.[33]. ResNet dirancang untuk mengatasi masalah hilangnya gradien yang sering terjadi pada *neural network* yang sangat dalam. Masalah ini coba diselesaikan oleh He et al. dengan memperkenalkan *framework deep residual learning*. Daripada mencoba mempelajari pemetaan langsung dari input ke output, ResNet bertujuan untuk mempelajari pemetaan residual atau perbedaan antara input dan output yang diinginkan. Dengan berfokus pada residual, *neural network* hanya perlu mempelajari penyimpangan atau perubahan yang diperlukan untuk mengubah input menjadi output. Konsep dari ResNet ini dapat digambarkan dengan persamaan :

$$\mathcal{F}(x) = \mathcal{H}(x) - x \tag{2.1}$$

Di mana $\mathcal{F}(x)$ adalah pemetaan residual atau perbedaan dari input dan output. He et al. berhipotesis bahwa akan lebih mudah untuk mengoptimalkan pemetaan residual daripada mengoptimalkan pemetaan asli yang tidak memiliki referensi sehingga persamaan (2.1) diformat menjadi :

$$\mathcal{H}(x) = \mathcal{F}(x) + x \tag{2.2}$$

Persamaan $\mathcal{F}(x) + x$ dapat direalisasikan dengan *feedforward neural network* dengan "*shortcut connections*"/jalan pintas. Di mana jalan pintas ini adalah pemetaan *identity* di mana output dari pemetaan ini akan di jumlahkan pada output lapisan bertumpuk atau pemetaan residual $\mathcal{F}(x)$. $\mathcal{H}(x)$ sendiri adalah output dari blok residual atau pemetaan yang diharapkan yang merupakan hasil penjumlahan dari informasi dari input asli x atau pemetaan *identity* dan pemetaan residual $\mathcal{F}(x)$. Gambar 2.11 merupakan ilustrasi dari persamaan (2.2) dan merupakan blok pembangun dari *residual network.*

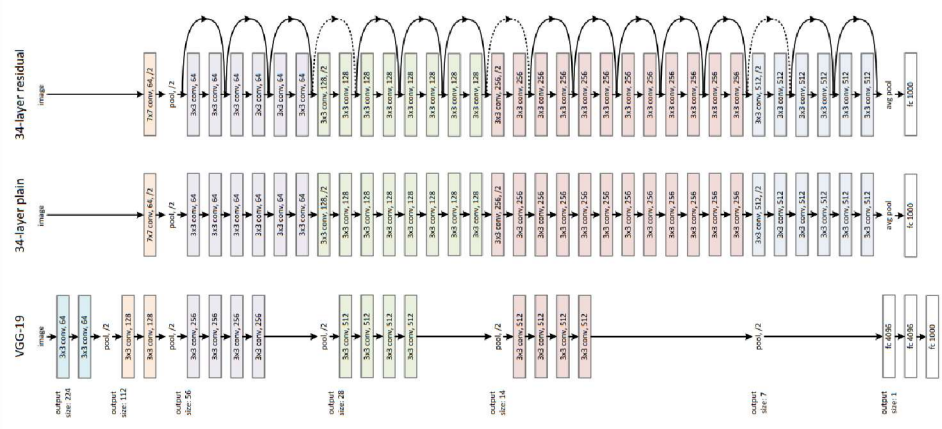


Gambar 2. 11 Blok Pembangun Residual Learning [33].

Arsitekrur ResNet terinspirasi oleh filosofi jaring VGG[62]. Di mana lapisan konvolusi pada umumnya menggunakan filter 3x3 dan mengikuti 2 aturan design sederhana:

1. Untuk ukuran peta fitur output yang sama, lapisan memiliki jumlah filter yang sama.
2. Jika ukuran peta fitur dibagi dua, jumlah filter digandakan untuk menjaga kompleksitas waktu per lapisan.

Downsampling dilakukan secara langsung dengan lapisan konvolusi yang memiliki *stride* sebanyak 2. *ResNet* diakhiri dengan lapisan *global average pooling* dan *fully-connected layer* 1000 dengan softmax dan *batch normalization* (BN) dilakukan tepat setelah setiap konvolusi dan sebelum aktivasi ReLU. Contoh arsitektur *ResNet* dapat dilihat pada Gambar 2.12.



Gambar 2. 12 Perbedaan arsitektur VGG-19 dan Resnet-34[33].

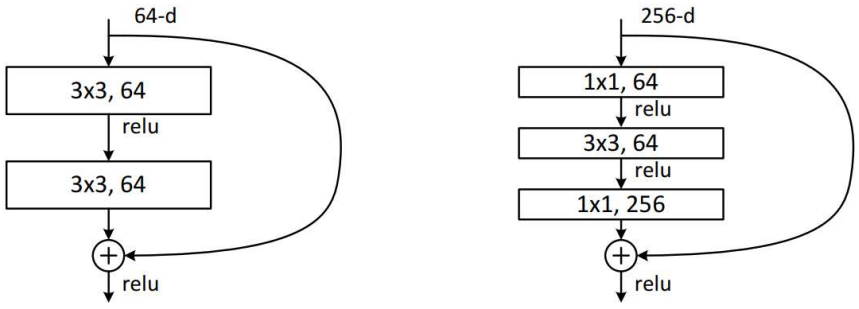
Berdasarkan arsitektur diatas *shortcut connections*/jalan pintas dimasukkan kedalam arsitektur tiap beberap lapisan / blok konvolusi. S*hortcut connections*

sendiri terbagi menjadi dua, *identity shortcut* dan *projection shortcut. Identity shortcut* digunakan ketika keuda input dan output lapisan memiliki dimensi yang sama, sedangkan *projection shortcut* digunakan ketika dimensi meningkat sehingga perlu dilakukan projeksi untuk menyamakan dimensi dari input dan output. Arsitektur *ResNet* dapat dilihat pada Tabel 2.2. Dapat dilihat blok konvolusi terdiri dari beberapa lapisan konvolusi.

Tabel 2. 2 Arsitektur ResNet 18-layer hingga 152-layer.

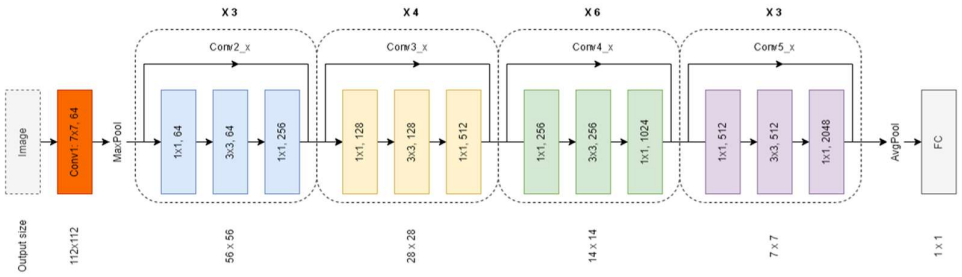
layer_name	output_size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112 x 112	7 x 7, 64, stride 2				
conv2_x	56 x 56	3 x 3, max pooling, stride 2				
		$\begin{bmatrix}3\times3 & 64\\ 3\times3 & 64\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3 & 64\\ 3\times3 & 64\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1 & 64\\ 3\times3 & 64\\ 1\times1 & 256\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1 & 64\\ 3\times3 & 64\\ 1\times1 & 256\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1 & 64\\ 3\times3 & 64\\ 1\times1 & 256\end{bmatrix}\times 3$
conv3_x	28 x 28	$\begin{bmatrix}3\times3 & 128\\ 3\times3 & 128\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3 & 128\\ 3\times3 & 128\end{bmatrix}\times 4$	$\begin{bmatrix}1\times1 & 128\\ 3\times3 & 128\\ 1\times1 & 512\end{bmatrix}\times 4$	$\begin{bmatrix}1\times1 & 128\\ 3\times3 & 128\\ 1\times1 & 512\end{bmatrix}\times 4$	$\begin{bmatrix}1\times1 & 128\\ 3\times3 & 128\\ 1\times1 & 512\end{bmatrix}\times 8$
conv4_x	14 x 14	$\begin{bmatrix}3\times3 & 256\\ 3\times3 & 256\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3 & 256\\ 3\times3 & 256\end{bmatrix}\times 6$	$\begin{bmatrix}1\times1 & 256\\ 3\times3 & 256\\ 1\times1 & 1024\end{bmatrix}\times 6$	$\begin{bmatrix}1\times1 & 256\\ 3\times3 & 256\\ 1\times1 & 1024\end{bmatrix}\times 23$	$\begin{bmatrix}1\times1 & 256\\ 3\times3 & 256\\ 1\times1 & 1024\end{bmatrix}\times 36$
conv5_x	7 x 7	$\begin{bmatrix}3\times3 & 512\\ 3\times3 & 512\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3 & 512\\ 3\times3 & 512\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1 & 512\\ 3\times3 & 512\\ 1\times1 & 2048\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1 & 512\\ 3\times3 & 512\\ 1\times1 & 2048\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1 & 512\\ 3\times3 & 512\\ 1\times1 & 2048\end{bmatrix}\times 3$
	1 x 1	average pool, 1000-d fc, softmax				

Untuk arsitektur *ResNet* yang lebih dalam khususnya *ResNet-50/101/152* diterapkan "*bottleneck building block*" di mana blok konvolusi dimofifikasi menjadi tiga lapisan dari dua lapisan sebelumnya. Tiga lapisan konvolusi tersebut ialah 1x1, 3x3, dan 1x1. Di mana lapisan 1x1 bertanggung jawab untuk mengurangi dan kemudian meningkatkan (memulihkan) dimensi, dan lapisan 3×3 sebagai *bottleneck* dengan dimensi input/output yang lebih kecil. Blok pembangun biasa dan *bottleneck* memiliki kompleksitas waktu yang sama. Blok pembangun biasa dan *bottleneck* dapat dilihat pada Gambar 2.13.

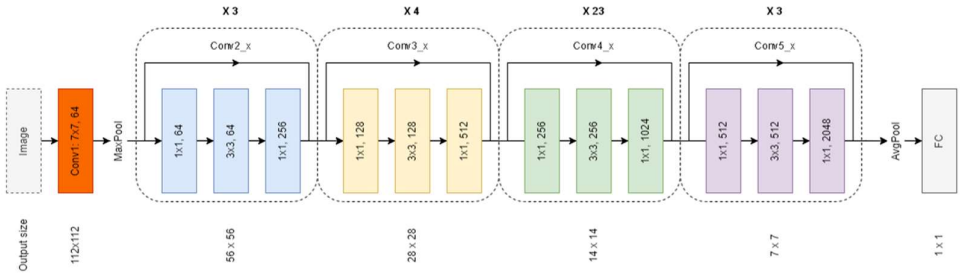


Gambar 2. 13 Blok pembangun ResNet (Kiri), blok pembangun bottleneck(Kanan)

Pada penelitian ini arsitektur ResNet-50 dan ResNet-101 yang akan digunakan. Ilustrasi ResNet-50 dan ResNet-101 dapat dilihat pada Gambar 2.14. dan Gambar 2.15.



Gambar 2. 14 Arsitektur ResNet-50.



Gambar 2. 15 Arsitektur ResNet-101.

2.2.8. Optimizer

Optimizer adalah algoritma atau metode yang digunakan untuk meminimalkan sebuah fungsi kesalahan (*loss function*). *Optimizer* biasanya berbentuk fungsi matematika dan berfungsi untuk menentukan bagaimana cara mengubah bobot dan *learning rate* dari *neural network* untuk mengurangi *loss function*[63]. Ada beberapa algoritma optimasi yang pada biasanya digunakan di antarannya[64][65]:

1. SGD atau *stochastic gradient descent,* adalah variasi dari *gradient descent* di mana gradien *loss function* di perkirakan dengan memilih secara acak subset contoh pelatihan, daripada menggunakan seluruh dataset pelatihan. Pengambilan sampel data yang *stochastic* atau secara acak ini memperkenalkan keacakan ke dalam proses pengoptimalan, yang dapat membantu algoritme keluar dari minimum lokal dan membuat konvergen

menjadi lebih cepat[63]. *SGD* memiliki *update rule/*aturan perubahan sebagai berikut[63]:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla F_i(\theta_t) \tag{2.3}$$

Dimana θ_t merepresentasikan parameter model (*weights* dan bias) pada iterasi ke-t. Kemudian α_t adalah *learning rate*, yang menentukan ukuran langkah untuk pembaruan parameter. Sedangkan $\nabla F_i(\theta_t)$ adalah gradien dari fungsi *loss* L sehubungan dengan parameter θ_t pada iterasi ke-t.

2. *Momentum*, adalah variasi dari *SGD* yang berusaha untuk menyelesai kan masalah di mana *SGD* mengalami kesulitan dalam menavigasi jurang, atau area di mana permukaan melengkung jauh lebih. *SGD* berosilasi melintasi lereng jurang sementara hanya membuat kemajuan yang ragu-ragu saja. *Momentum* membantu mempercepat *SGD* ke arah yang relevan dan meredam osilasi, ini dilakukan dengan menambahkan pecahan/*fraction "γ"* dari vektor pembaruan dari langkah waktu sebelumnya ke vektor pembaruan saat ini. *Momentum* memiliki *update rule/*aturan perubahan sebagai berikut[64]:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta \mathrm{J}(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{2.4}$$

v_t adalah istilah momentum yang diperbarui pada iterasi t. Bagian γv_{t-1} merepresentasikan akumulasi momentum dari iterasi sebelumnya, disesuaikan dengan koefisien momentum γ. Untuk simbol η merepresentasikan *learning rate*, yang menentukan ukuran langkah untuk pembaruan parameter. Bagian $\nabla_\theta \mathrm{J}(\theta)$ adalah gradien dari fungsi *loss* J sehubungan dengan parameter θ pada iterasi t saat ini. Kemudian pada persamaan $\theta = \theta - v_t$ parameter model diperbarui dengan mengurangi suku momentum. Hal ini memiliki efek memindahkan parameter ke arah yang memperhitungkan arah gradien saat ini dan arah akumulasi dari iterasi sebelumnya.

3. *Adam* atau *Adaptive Moment Estimation*, adalah sebuah algoritma optimasi yang menggabungkan RMSprop dan Momentum. *Adam* melakukan perhitungan *learning rate* yang adaptif untuk setiap parameter. Selain

menyimpan rata-rata *decay* eksponensial dari gradien kuadrat masa lalu v_t seperti *Adadelta* dan *RMSprop*, *Adam* juga menyimpan rata-rata *decay* eksponensial dari gradien masa lalu m_t, mirip dengan *optimizer momentum*:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2.5)$$

Di mana m_t dan v_t masing-masing adalah perkiraan dari momen pertama (rata-rata) dan momen kedua (varians tidak terpusat) dari gradien. Pada m_t bagian $\beta_1 m_{t-1}$ menentukan berapa banyak estimasi momen pertama sebelumnya m_{t-1} yang dipertahankan dalam estimasi saat ini. Di mana β_1 adalah tingkat peluruhan eksponensial untuk estimasi momen pertama. Dan m_{t-1} adalah estimasi momen pertama dari iterasi sebelumnya (t-1). Bagian $(1 - \beta_1) g_t$ adalah kontribusi gradien saat ini terhadap pembaruan estimasi momen pertama. Di mana $1 - \beta_1$ berfungsi sebagai bobot untuk suku gradien saat ini, dan g_t adalah gradien dari fungsi *loss* sehubungan dengan parameter model pada iterasi ke-t.

Pada v_t bagian $\beta_2 v_{t-1}$ adalah kontribusi dari estimasi momen kedua sebelumnya terhadap pembaruan estimasi momen kedua saat ini. Di mana β_2 adalah tingkat peluruhan eksponensial untuk estimasi momen kedua, dan v_{t-1} adalah estimasi momen kedua dari iterasi sebelumnya (t-1). Bagian $(1 - \beta_2) g_t^2$ adalah kontribusi gradien saat ini terhadap pembaruan estimasi momen kedua. Di mana $1 - \beta_2$ berfungsi sebagai bobot untuk suku gradien saat ini, dan g_t^2 adalah vektor gradien kuadrat berelemen dari fungsi *loss* pada iterasi ke-t.

Karena m_t dan v_t diinisialisasi sebagai vektor 0, penulis *Adam* mengamati bahwa mereka condong bias ke nol, terutama selama langkah waktu awal, dan terutama ketika laju peluruhan kecil (yaitu β1 dan β2 mendekati 1). Bias ini dinetralkan dengan menghitung estimasi momen pertama dan kedua yang sudah terkoreksi bias:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (2.6)$$

Untuk medapatkan $\hat{m}_t$ persamaan ini membagi estimasi momen pertama mentah m_t dengan $1-\beta_1^t$. Di mana m_t adalah estimasi momen pertama mentah dari gradien pada iterasi t. $1-\beta_1^t$ adalah faktor yang berubah seiring dengan jumlah iterasi t. Faktor ini diperkenalkan untuk mengoreksi bias pada estimasi awal saat pertama.

Kemudian untuk medapatkan $\hat{v}_t$ persamaan ini membagi estimasi momen kedua mentah v_t dengan $1-\beta_2^t$. Di mana v_t adalah estimasi momen kedua mentah dari gradien pada iterasi t. $1-\beta_2^t$ adalah faktor yang berubah seiring dengan jumlah iterasi t. Faktor ini diperkenalkan untuk mengoreksi bias pada estimasi awal saat kedua.

$\hat{m}_t$ dan $\hat{v}_t$ kemudian digunakan untuk memperbarui parameter, yang menghasilkan *update rule/*aturan perubahan *Adam*[66]:

$$\theta_{t-1} = \theta_t \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \quad (2.7)$$

Dimana θ_t merepresentasikan parameter model pada iterasi t saat ini. η adalah *learning rate*, yang menentukan ukuran langkah untuk pembaruan parameter. $\sqrt{\hat{v}_t}+\epsilon$ adalah akar kuadrat dari estimasi momen kedua yang dikoreksi bias, dan ϵ adalah konstanta kecil yang ditambahkan untuk stabilitas numerik. Ini mencegah pembagian dengan nol. $\hat{m}_t$ merupakan estimasi momen pertama yang dikoreksi bias dari gradien pada iterasi ke-t. Ini merupakan rata-rata bergerak dari gradien masa lalu dengan koreksi bias.

2.2.9. Bounding Box Regression

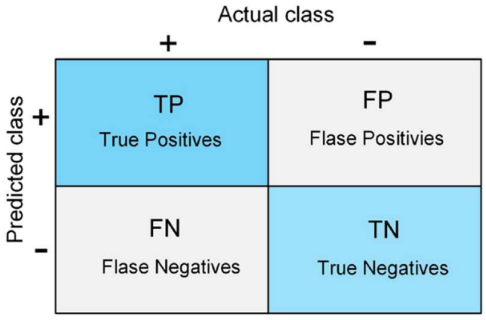
Deteksi objek adalah masalah pembelajaran *multitask* yang terdiri dari lokalisasi objek dan klasifikasi objek. Kedua sub tugas tersebut selalu menjadi salah satu masalah paling mendasar dalam *computer vision*. Detektor objek bergantung pada *bounding box regression* (BBR) untuk melokalisasi objek[67]. *Bounding box regression* merupakan langkah penting dalam pendeteksian objek. Hal ini bertujuan

untuk memperbaiki lokasi kotak pembatas yang diprediksi berdasarkan proposal awal atau *anchor box*[68]. *Bounding box* berisi empat koordinat dari sebuah objek, empat koordinat tersebut merepresentasikan dua titik pada kiri atas (x1, y1) dan kanan bawah (x2, y2) dari kotak seperti yang dapat dilihat di Gambar 2.16. Tugas dari *bounding box regression* adalah untuk meregresi kedua titik koordinat dari *bounding box* untuk target objek, dengan menggunakan *regressor*[69].

Gambar 2. 16 Ilustrasi Bounding Box.

2.2.10. Confusion Matrix

Confusion Matrix merupakan salah satuh metode pengukuran keputusan dalam *supervised machine learning*[70]. *Confusion Matrix* berisikan informasi-informasi tentang hasil aktual dan prediksi dari sistem klasifikasi. Metode ini memvisualisasikan tingkat kebingungan algoritme di dalam kelas-kelas yang berbeda dan tidak bergantung pada algoritme klasifikasi konkret[71].



Gambar 2. 17 Confusion Matrix[72].

Pada Gambar 2.17 *True Positives* (TP) dan *True Negatives* (TN) menandakan jumlah kelas positif dan negatif yang berhasil dikategorikan secara benar. Sedangkan *False Positives* (FP) dan *True Negatives* (TN) menandakan jumlah kelas positif dan negatif yang tidak berhasil dikategorikan secara benar[73].

Sejumlah ukuran kinerja klasifikasi dapat didefinisikan dari empat hasil klasifikasi ini. Beberapa di antaranya yaitu[74][75]:

a. Akurasi

Akurasi adalah salah satu metrik yang sering dipakai dalam evaluasi kinerja model klasifikasi *machine learning*. *Confusion matrix* menghitung rasio jumlah klasifikasi benar dan total jumlah sampel dalam set data evaluasi. Tetap akurasi sebagai *metric* dapat menyesatkan dalam kasus proporsi kelas yang berbeda karena hanya dengan menetapkan semua sampel ke kelas yang prevalen adalah cara mudah untuk mencapai akurasi yang tinggi. Akurasi dapat dicari menggunakan persamaan (2.8).

$$Akurasi = \frac{TP + TN}{TP + FN + TN + FP} \tag{2.8}$$

b. Presisi

Presisi adalah metrik yang mengukur proporsi *True Positives* yang benar (yaitu jumlah prediksi positif yang benar) dari semua prediksi positif yang dibuat oleh model. Dapat di katakan presisi mengukur berapa banyak prediksi positif yang dibuat oleh model yang benar-benar positif. Presisi adalah metrik evaluasi yang baik untuk digunakan ketika nilai *false positive* sangat tinggi dan nilai *false negative* rendah. Presisi dapat dicari menggunakan persamaan (2.9).

$$Presisi = \frac{TP}{TP + FP} \tag{2.9}$$

c. Recall

Recall, juga dikenal sebagai sensitivitas atau *True Positive Rate* (TPR), adalah metrik yang mengukur tingkatan sampel positif yang diklasifikasikan dengan benar. *Recall* dihitung sebagai rasio antara sampel positif yang diklasifikasikan dengan benar dari keseeluruhan sampel positif yang sebenarnya ada. *Recall* dapat dicari menggunakan persamaan (2.10).

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

d. F- Measure

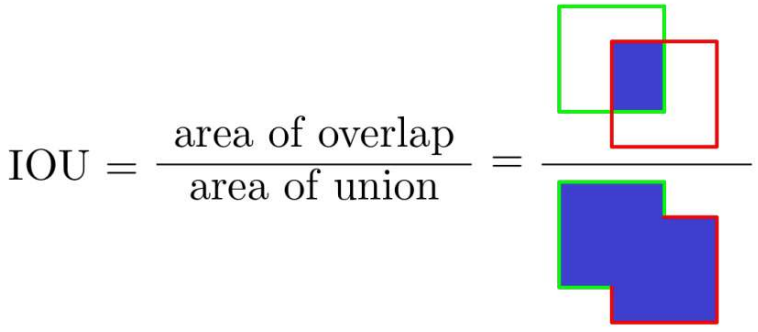
F-measure, juga dikenal sebagai *F1-score*, adalah metrik dalam *machine learning* yang menggabungkan presisi dan *recall* ke dalam satu metrik. *F-measure* adalah harmonisasi dari rata-rata presisi dan *recall*, di mana nilai terbaiknya adalah 1,0 dan yang terburuk adalah 0,0. *F-Measure* dapat dicari menggunakan persamaan (2.11).

$$F1\ Score = 2\frac{Presisi * Recall}{Presisi + Recall} \tag{2.11}$$

2.2.11. Intersection over union (IOU)

Pertimbangkan objek target deteksi yang diwakili oleh *bounding box ground-truth* B_{gt} dan area yang terdeteksi yang diwakili oleh *bounding box* prediksi B_p. Tanpa harus mempertimbangkan tingkat kepercayaan, dapat disimpulkan kecocokan yang sempurna adalah ketika area dan lokasi dari kotak prediksi dan ground-truth sama atau saling tumpeng tindih. Kedua kondisi ini dinilai dengan *intersection over union* (IOU), sebuah pengukuran yang didasarkan pada Indeks *Jaccard*, sebuah koefisien kesamaan untuk dua set data. Dalam lingkup deteksi objek, IOU sama dengan area tumpang tindih/*overlap* (persimpangan/*intersection*) antara *bounding box* prediksi B_p dan *bounding box ground-truth* B_{gt} dibagi dengan area penyatuannya. Diilustrasikan pada Gambar 2.18, dan di repesentasikan pada persamaan (2.12) yaitu[76]:

$$J(B_p, B_{gt}) = IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{2.12}$$



Gambar 2. 18 Ilustrasi dari Intersection over union (IOU)[76].

2.2.12. Precision dan Recall pada Objek Deteksi

Precision adalah kemampuan model untuk mengidentifikasi hanya objek yang relevan. Ini dihitung sebagai persentase prediksi positif yang benar dari seluruh prediksi positif yang dibuat oleh model. Sedangkan *Recall* adalah kemampuan model untuk mencari semua kasus-kasus relevan atau semua *bounding box* objek yang sebenarnya. *Recall* dihitung sebagai persentase prediksi positif yang benar di antara semua objek yang sebenarnya ada *(ground truth)*. Untuk menghitung nilai *precision* dan *recall*, setiap *bounding box* yang terdeteksi harus diklasifikasikan terlebih dahulu menjadi[77]:

- *True positive* (TP): Deteksi yang benar atau *bounding box* sesuai dengan *ground truth*/kenyataan;
- *False positive* (FP): Deteksi yang salah atau objek relevan tidak ada dalam *bounding box*;
- *False negative* (FN): *Bounding box ground truth* yang tidak terdeteksi.

Dengan asumsi ada dataset dengan *ground-truth* "G" dan model yang menghasilkan deteksi "N", di mana "S" adalah benar (S ≤ G), konsep *precision* dan *recall* dapat secara formal dinyatakan sebagai[76]:

$$Pr = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{N-S} FP_n} = \frac{\sum_{n=1}^{S} TP_n}{all\ detection} \tag{2.13}$$

$$Rc = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{G-S} FN_n} = \frac{\sum_{n=1}^{S} TP_n}{all\ ground\ truth} \tag{2.14}$$

2.2.13. Precision Recall Curve

Precision × *recall curve* dapat dilihat sebagai pertukaran antara presisi dan *recall* untuk nilai kepercayaan yang berbeda yang terkait dengan kotak pembatas yang dihasilkan oleh detektor. Jika keyakinan detektor sedemikian rupa sehingga FP-nya rendah, maka presisi akan tinggi. Namun, dalam hal ini banyak positif yang mungkin terlewatkan, menghasilkan FN yang tinggi, dan dengan demikian menghasilkan *recall* yang rendah. Sebaliknya, jika seseorang menerima lebih banyak positif, *recall* akan meningkat, tetapi FP juga dapat meningkat, sehingga menurunkan presisi. Namun, detektor objek yang baik harus menemukan semua objek yang benar (FN = 0 ≡ *recall* tinggi) sambil mengidentifikasi hanya objek yang

relevan (FP = 0 ≡ presisi tinggi). Oleh karena itu, detektor objek tertentu dapat dianggap baik jika presisi tetap tinggi seiring dengan meningkatnya *recall*, yang berarti bahwa jika ambang batas/*threshold* kepercayaan bervariasi, presisi dan recall akan tetap tinggi. Oleh karena itu, area yang tinggi di bawah kurva (AUC) cenderung menunjukkan presisi dan *recall* yang tinggi[77]. Sehingga *precision recall curve* dapat digunakan sebagai metic pembanding antara model objek deteksi.

2.2.14. Average Precision

Average precision (AP) adalah metrik yang didasarkan pada area di bawah *precision* × *recall curve* yang telah diproses sebelumnya untuk menghilangkan perilaku zig-zag. Metrik ini meringkas pertukaran *precision-recall* yang ditentukan oleh tingkat kepercayaan dari kotak pembatas yang diprediksi. Untuk menghitung AP, terlebih dahulu nilai kepercayaan K tiap deteksi harus di urutkan dari yang tertinggi ke terendah. Kemudian pasangan *precision* × *recall* harus diinterpolasi sedemikian rupa sehingga kurva *precision* × *recall* yang dihasilkan adalah monoton. AP di hitung dengan mengambil sampel interpolasi dari *precision* × *recall* pada nilai referensi *recall*[77]. Ada dua pendekatan untuk melakukan interpolasi, yaitu[76]*:*

1. *n-point interpolation:* Dalam *n-point interpolation*, himpunan nilai *recall* $R_r(n)$ untuk komputasi memiliki jarak yang sama dalam interval [0, 1] di mana:

   $$R_r(n) = \frac{N-n}{N-1}, n = 1, 2, \dots, N. \tag{2.15}$$

   Sehingga persamaan AP untuk *n-point interpolation* adalah :

   $$AP = \frac{1}{N}\sum_{n=1}^{N} Pr_{interp}\left(R_r(n)\right) \tag{2.16}$$

   Aplikasi populer dari metode interpolasi ini menggunakan N = 101 dan N = 11.

2. *all-point interpolation:* Untuk perhitungan AP menggunakan *all-point interpolation* atau AP_{all}, himpunan nilai recall yang digunakan sama persis dengan set nilai recall yang dihitung dengan mempertimbangkan semua tingkat kepercayaan, dengan tingkat kepercayaan $\tau(0) = 0$ dan

τ(K+1) = 1 yang ditambahkan sehingga titik-titik Rr(0) = 1 dan Rr(K+1) = 0. Lebih tepatnya digambarkan pada persamaan di bawah,

$$R_r(0) = 1$$
$$R_r(k) = Rc\big(\tau(k)\big), k = 1, 2, \dots, K, \tag{2.17}$$
$$R_r(K+1) = 0$$

Sehingga persamaan AP untuk *all-point interpolation* adalah :

$$AP = \sum_{k=0}^{K} (R_r(k) - R_r(k+1)) Rr_{interp}\,(R_r(k)) \tag{2.18}$$

2.2.15. Mean Average Precision

Mean Average Precision atau *mAP* adalah metrik yang digunakan untuk mengukur akurasi suatu detektor/model objek deteksi. *Mean Average Precision* mempertimbangkan semua kelas dalam basis data tertentu. *Mean Average Precision* didapatkan dengan mencari rata-rata dari seluruh *Average Precision* (AP) dari setiap kelas. *Mean Average Precision* dapat dihitung dengan persamaan berikut[76],

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.19}$$

Di mana N adalah jumlah kelas, dan AP_i adalah AP untuk tiap kelas "i".

2.2.16. TensorFlow

TensorFlow adalah platform *open source end-to-end* untuk *machine learning*. *Tensorflow* memiliki ekosistem *tools*, pustaka, dan sumber daya komunitas yang komprehensif dan fleksibel yang memungkinkan para peneliti mendorong kemajuan dalam *machine learning.* Dengan tujuan membantu pengembang agar dapat dengan mudah membangun dan menggunakan aplikasi bertenaga *machine learning*[78]. Hal ini memungkinkan pengembang untuk membuat dan melatih model menggunakan berbagai algoritme yang berbeda, termasuk *convolutional neural network*, *recurrent neural network*, *deep neural network*, dan banyak lainnya. *TensorFlow* juga menyediakan berbagai alat untuk

memvisualisasikan dan *men-debug* proses pelatihan, serta alat untuk men-deploy model yang telah dilatih[79].

2.2.17. TensorFlow Object Detection API

TensorFlow Object Detection API adalah sebuah *framework open source* yang dibangun diatas *tensorflow* yang bertujuan untuk memudahkan membangun, melatih, dan menerapkan model *object detection*. *TensorFlow Object Detection API* menyediakan user dengan model *pre-trained* yang disertai dengan instruksi dan contoh penerapan untuk melakukan *fine-tuning* dan objek deteksi[80]. Pada penelitian ini model *Faster R-CNN* dengan *ResNet-50* dan *ResNet-101* dipilih.