SISTEM DETEKSI PERGERAKAN MANUSIA ATAU OBJEK PENTING DARI REKAMAN KAMERA PENGAWAS MENGGUNAKAN *FASTFLOWNET* DAN *YOLOv3*

TUGAS AKHIR

Diajukan sebagai syarat menyelesaikan jenjang strata Satu (S-1)
di Program Studi Teknik Informatika, Jurusan Teknologi,
Produksi dan Industri, Institut Teknologi Sumatera

Oleh:

Muhammad Ariefuddin Satria Dharma

119140149

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI, PRODUKSI DAN INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN

2022

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR RUMUS

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Berdasarkan perhitungan Badan Pusat Statistik Indonesia dalam 1 menit 32 detik terjadi 1 tindakan kriminal di Indonesia. Hal ini merujuk kepada penelitian yang dilakukan oleh Khairani dan Ariesa [1], yang menyebutkan 140 dari 100.000 orang di Indonesia berisiko terkena tindak kejahatan. Berbagai macam cara sudah dilakukan untuk menangani tindakan kriminal, salah satunya dengan memasang kamera pengawas. Berdasarkan penelitian yang dilakukan oleh Khan, dkk. [2], dan Jansen, dkk. [3]. Kamera pengawasan ini memiliki peran penting dalam mengurangi tingkat kejahatan dan meningkatkan rasa aman bagi masyarakat yang diawasi oleh kamera pengawas. Hal ini menyebabkan kamera pengawas marak digunakan oleh masyarakat [4], namun terdapat beberapa masalah pada penggunaan kamera pengawas ini.

Salah satu masalahnya adalah kapasitas penyimpanan video rekaman dari kamera pengawas, yang dimana hal ini bisa menjadi masalah besar karena kamera pengawas yang beroperasi selama 24 jam setiap harinya. Berdasarkan data yang pada *video surveillance storage matrix* [5] buatan *seagate* dapat disimpulkan bahwa untuk rekaman video dengan MPEG-4 *encoding*, resolusi 704 x 480 *pixel* dan *frame rate* 20fps *hard drive* dengan kapasitas 1 *Tera Byte* akan penuh dalam waktu 42 hari. Sedangkan, video dengan *encoding* H.264, dengan resolusi 704 x 480 dan *frame rate* 20fps hard drive dengan kapasitas 1 *Tera Byte* akan penuh dalam waktu 44 hari. Hal ini dapat menyebabkan pemilik kamera pengawas harus melakukan *backup* atau pergantian *hard drive* setidaknya 1 bulan sekali yang dimana hal ini akan berimbas pada biaya operasional kamera pengawas tersebut.

Terdapat dua solusi yang dapat digunakan untuk menyelesaikan masalah ini. Solusi pertama, pemilik kamera pengawas bisa mengawasi kamera pengawas secara langsung. Akan tetapi, solusi ini akan memiliki hasil yang kurang konsisten. Hal ini disampaikan di penelitian [6] yang telah membuktikan bahwa setiap orang memiliki

tingkat kewaspadaan yang berbeda-beda, menyebabkan hasil pengawasan yang tidak konsisten. Solusi kedua adalah dengan menggunakan sistem berbasis *Artificial Intelligence* atau lebih tepatnya *Computer Vision*. Sistem ini menurut beberapa penelitian [7], [8], [9] merupakan sistem yang menjanjikan serta memiliki performa yang lebih baik dibandingkan metode yang pertama.

Computer Vision merupakan bidang yang berusaha membuat sistem yang dapat mendapatkan data atau informasi dari gambar [10]. Hal ini dapat dilakukan dengan cara mencari properti yang gambar tersebut miliki seperti warna dan bentuk. Metode ini pula yang nantinya akan dijadikan dasar untuk mengklasifikasikan atau mendeteksi objek dengan menggunakan model *Machine Learning* atau *Deep Learning*. Metode *Deep Learning* merupakan subset dari *Machine Learning* yang dimana metode ini terinspirasi dari cara otak manusia memproses informasi. Metode ini tidak membutuhkan aturan yang detail dari manusia akan tetapi lebih membutuhkan data dengan kuantitas yang besar [11]**.**

Berdasarkan hasil dari beberapa penelitian sebelumnya, penulis menemukan beberapa algoritma yang diimplementasikan atau digunakan sebagai pembanding dari algoritma yang diajukan pada penelitian deteksian objek dan gerakan yaitu metode FastFlowNet [12] dan juga YOLO [13]. Dimana YOLO adalah algoritma pendeteksian objek [14] sedangkan FastFlowNet merupakan algoritma yang digunakan untuk mendeteksi pergerakan [15]. Dengan menggunakan kedua algoritma diatas, penulis berharap bisa membuat sistem yang menggabungkan kedua algoritma tersebut dengan tujuan memperkecil durasi *video* rekaman kamera pengawas.

Beberapa *metrics* digunakan untuk memastikan performa sistem, penulis akan menggunakan *metrics* perhitungan *precision*, *recall* dan juga *F1-score* untuk menilai kelayakan sistem yang dikembangkan. *Metrics* ini akan menjelaskan bagaimana sistem memilah bagian-bagian pada video yang memiliki pergerakan objek penting dan mana bagian-bagian yang tidak.

Oleh karena itu penulis bermaksud untuk mengangkat penelitian yang berjudul **"SISTEM DETEKSI PERGERAKAN MANUSIA ATAU OBJEK PENTING DARI REKAMAN KAMERA PENGAWAS MENGGUNAKAN FASTFLOWNET DAN YOLOv3"** untuk membantu meningkatkan efisiensi ruang penyimpanan video rekaman dengan menggunakan bahasa pemrograman Python serta dibantu dengan *computer vision library* yaitu OpenCV dan juga *Deep Learning Framework* Pytorch.

1.2 Rumusan Masalah

Berdasarkan Latar Belakang yang telah dipaparkan oleh penulis di sub-bab yang 1.1 maka didapatkan rumusan masalah yaitu :

1. Bagaimana rancangan sistem *computer vision* untuk melakukan segmentasi *video* rekaman kamera pengawas dengan melibatkan proses deteksi pergerakan dan deteksi objek dengan algoritma FastFlowNet dan juga YOLOv3?.
2. Bagaimana performa yang didapatkan dari sistem hasil penggabungan algoritma FastFlowNet dan juga YOLOv3?

1.3 Tujuan Penelitian

Berdasarkan Latar Belakang dan Rumusan Masalah yang telah dipaparkan penulis di sub-bab 1.1 dan 1.2, maka didapat tujuan penelitian sebagai berikut :

1. Merancang sistem *Computer Vision* dengan menerapkan algoritma FastFlowNet untuk melakukan deteksi pergerakan dan algoritma YOLOv3 untuk melakukan pendeteksian pegerakan manusia dan objek penting.
2. Dihasilkan sistem yang mampu memberikan segmentasi *video* melalui proses lokalisasi pergerakan manusia dan objek penting, dan menghapus bagian *video* lainnya yang tidak penting.

1.4 Batasan Masalah

Adapun Batasan Masalah yang diterapkan pada penelitian yang dilakukan penulis sebagai berikut:

1. Sistem akan menerima masukan berupa *video* rekaman yang terdapat tindakan kriminalitas.

2. Keluaran dari sistem adalah *video* yang sudah tersegmentasi dengan fokus melokalisasi suatu pergerakan yang melibatkan objek penting.
3. Penelitian menggunakan *dataset* yang biasa digunakan untuk melakukan perhitungan model *object detection* (*UCF Crime Dataset*).
4. Proses pendeteksian berfokus pada aktifitas-aktifitas seperti Perampokan, Pembegalan, dan Pencurian yang diambil dari kumpulan data *UCF Crime*.
5. Penelitian menggunakan *pre-trained* model dari FastFlowNet dan YOLOv3.
6. *Dataset* video yang digunakan memiliki durasi maksimal 10 menit.

1.5 Manfaat Penelitian

Adapun Manfaat yang didapat dari hasil penelitian yang dilakukan oleh penulis adalah sebagai berikut :

1. Sistem yang dihasilkan dari penelitian ini dapat melokalisasi pergerakan penting, dengan harapan untuk memperkecil besar video rekaman yang perlu disimpan oleh *hard drive*.
2. Sistem yang dihasilkan dari penelitan ini dapat memperkecil biaya oprasional dari kemera pengawas.

1.6 Sistematika Penulisan

Sistematika penulisan berisi pembahasan apa yang akan ditulis di setiap Bab. Sistematika pada umumnya berupa paragraf yang setiap paragraf mencerminkan bahasan setiap Bab.

1.6.1 BAB I PENDAHULUAN

Pada bagian ini penulis akan membahas tentang Latar Belakang dari penelitian ini, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian dan juga sistematika penulisan.

1.6.2 BAB II TINJAUAN PUSTAKA

Pada bab ini membahas tentang tinjauan pustaka penelitian terdahulu yang terkait dengan penelitian yang akan penulis lakukan.

1.6.3 BAB III METODE PENELITIAN

Pada bab ini berisikan pedoman tentang tahapan - tahapan yang dilakukan pada penelitian ini agar hasil dari penelitian ini tidak menyimpang dari tujuan penelitian yang telah ditentukan sebelumnya.

BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian terkait dengan *Motion Detection* dan juga *Object Detection* dikaji dan dipelajari guna dijadikan acuan dan pembanding bagi penulis saat mengerjakan penelitian ini. Penelitian-penelitian yang dijadikan acuan dan pembanding untuk penelitian penulis dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

No	Judul	Metode	Dataset	Hasil
1	MobileNetV2: Inverted Residuals and Linear Bottlenecks [16]	SSD MobileNet V2	COCO	Mendapatkan Algoritma yang memiliki interference time sebesar 75ms pada Google Pixel 1 dan AP (Average Precision) sebesar 22.1%.
2	Comparative study of motion detection methods for video surveillance systems [17]	*Classical Computer Vision Method*	CDnet video dataset	Tidak ada metode yang bisa baik untuk semua kasus, hanya saja, penulis merekomendasikan Self-Organized Background Subtraction (Simp-SOBS), dan juga Running Gaussian average (RGA). Dengan presisi sebesar 51.477% dan 49.4%.

No	Judul	Metode	Dataset	Hasil
3	Motion Detection and Classification: ultra-fast road user detection [13]	YOLOv3, TinyYOLO, Gaussian Mixture Model + ResNet18	Camera Recording Aalto University 1280 x 720p H264	Hasil dari perbandingan antara Algoritma YOLOv3, TinyYOLO, serta algoritma gabungan Gaussian Mixture model dan ResNet18 didapatkan Presisi terbesar adalah metode YOLOv3 dengan mAP 60.6%.
4	Optical Flow Based Motion Detection for Autonomous Driving [12]	FastFlowNet, RAFT	nuScenes	Hasil dari perbandingan antara Algoritma RAFT dan FastFlowNet dengan Presisi terbesar didapat dari metode FastFlowNet dengan presisi 94.3%
5	YOLOv3: An Incremental Improvement [18]	YOLOv3	COCO	Didapatkan sebuah algortima YOLO versi ke 3 yang dapat melakukan pendeteksian *bounding box* pada objek kecil dengan baik dengan mAP yang didapatkan adalah 33%

No	Judul	Metode	Dataset	Hasil
6	Comparison of Object Detection Algorithms for Street-level Objects [19]	SSD MobileNetv2 FPN-Lite, YOLOv3, YOLOv4, YOLOv5l, YOLOv5s	Modified Udacity Self Driving Car Dataset.	Hasil perbadingkan dari metode MobileNet dan YOLO adalah Algoritma YOLO mendapatkan mAP@.5 setidaknya 0.5 mAP.

Banyak penelitian telah dilakukan yang berfokus pada masalah pendeteksian objek menggunakan metode *Deep Learning* sejak munculnya AlexNet pada tahun 2012. Pada penelitian yang diajukan oleh Sandler, dkk.[16], mengusulkan algoritma MobileNetV2 untuk membuat model *Computer Vision* untuk *mobile device* atau *device* yang memiliki keterbatasan *resource*. Penelitian ini menggunakan dataset COCO yang dimana setelah dilakukan pengujian didapatkan nilai sebesar 22.1% mAP [16].

Tidak hanya algoritma MobileNetV2 saja yang diusulkan pada bidang deteksi objek. algoritma YOLOv3 yang diajukan oleh Joseph Redmon dan Ali Farhadi [18]. Pada penelitiannya, Joseph Redmon dan Ali Farhadi mengusulkan algoritma YOLOv3 untuk membuat performa model jadi lebih baik dari versi sebelumnya. Dengan cara mencoba menggabungkan metode-metode yang didapatkan peneliti dari penelitian lain [18]. Hasil dari penelitian ini adalah algoritma *object detection* yang memiliki mAP 33%.

Percobaan terhadap YOLOv3 dilakukan di beberapa penelitian. Salah satunya, penelitian yang dilakukan oleh Ojala, dkk.[13], penelitian tersebut mengusulkan algoritma *Computer Vision* untuk melakukan pendeteksian dan lokalisasi objek. Dengan menggabungkan *Gaussian Mixture Model(GMM)* untuk mendeteksi objek yang bergerak dan juga ResNet18 untuk klasifikasi objek tersebut [13]. Algoritma ini dibandingkan dengan algoritma YOLOv3 dan juga TinyYOLO dengan hasil nilai mAP sebesar 37.1% yang lebih besar dibandingkan nilai mAP TinyYOLO yang hanya 27.3%. Akan tetapi, lebih kecil dibandingkan nilai mAP algoritma YOLOv3 yaitu 60.6% [13].

Pada penelitian yang dilakukan oleh Naftali, dkk. [19], beberapa versi algoritma YOLO dibandingkan dengan algoritma SSD MobileNetV2 untuk mendeteksi objek di jalan menggunakan *Udacity*'s *Self Driving Car Dataset*. Didapatkan bahwa metode MobileNetV2 mendapatkan nilai mAP sebesar 0.315. Lebih kecil dibandingkan model-model yang menggunakan algoritma YOLO yang memiliki nilai mAP@.5 terkecil 0.530 dengan perbandingan yang tidak terlalu jauh antara semua versi YOLO.

Pada sisi lain, telah dilakukan beberapa penelitian yang berkaitan dengan pendeteksian gerak pada sebuah citra. Pada penelitian yang dilakukan oleh Sehairi, dkk. [17], dilakukan perbandingan metode-metode *Computer Vision* untuk mendeteksi pergerakan diantaranya metode *temporal differencing* (*frame difference*), *three frame difference* (3FD), *adaptive background* (average filter), *forgetting morphological temporal gradient* (FMTG), ∑Δ *Background estimation*, *spatio-temporal Markov field*, *running Gaussian average* (RGA), *mixture of Gaussians* (MoG), *spatio-temporal entropy image* (STEI), *difference-based spatio-temporal entropy image* (DSTEI), *eigen-background* (Eig-Bg) dan *simplified self-organized map* (Simp-SOBS). Penelitian ini menyimpulkan bahwa metode metode yang dibandingkan tidak bisa berjalan dengan baik untuk setiap kasus yang diuji [17],.

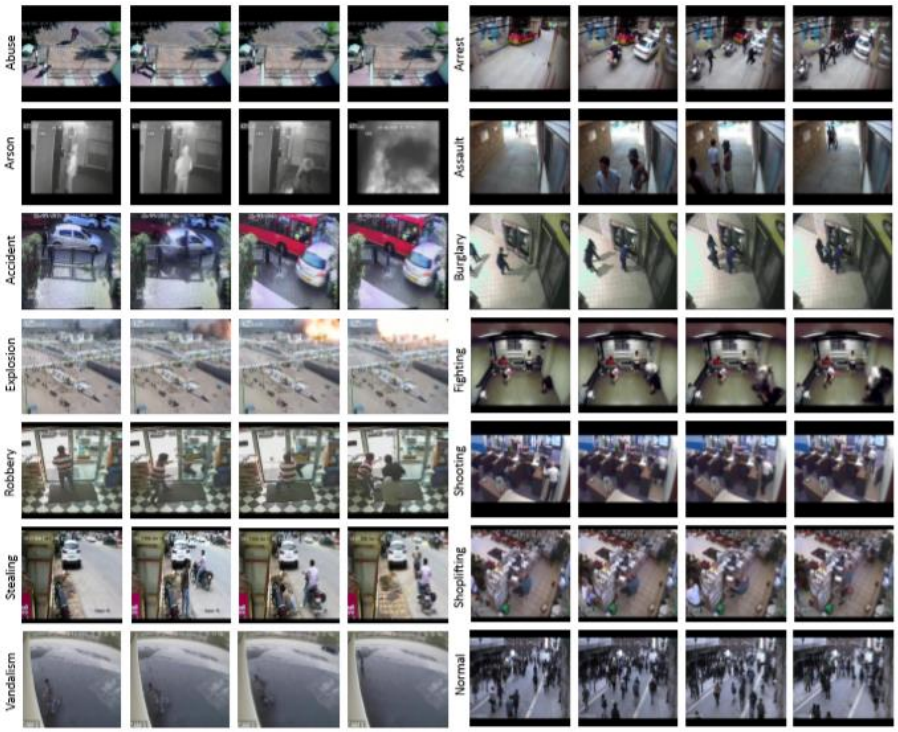
Terdapat pula penelitian pendeteksian gerakan yang menggunakan metode *Deep Learning*. Seperti pada penelitian yang dilakukan oleh Ka Man Lo [12]. Pada penelitian ini dilakukan perbandingan pada model algoritma *Optical Flow Estimation* untuk mendeteksi apakah mobil bergerak atau tidak. Algoritma yang digunakan pada penelitan ini yaitu algoritma FastFlowNet, dan RAFT yang digabungkan dengan algoritma ResNet18 yang bertugas melakukan prediksi apakah mobil bergerak atau tidak. Hasil dari penelitian adalah model dengan algoritma FastFlowNet mendapatkan nilai presisi sebesar 94.3% [12]. Sedangkan, untuk model dengan algoritma RAFT didapatkan nilai presisi sebesar 89.9% [12].

Berdasarkan penelitian-penelitian tentang *Object Detection* dan juga *Motion Detection* yang telah dilakukan. Penulis memutuskan akan melakukan penelitian dengan menggabungkan metode FastFlowNet dan juga YOLOv3 dalam masalah pendeteksian pergerakan objek-objek penting.

2.2. Dasar Teori

Dalam penelitian tugas akhir diperlukan teori pendukung lebih lanjut agar peneliti mendapatkan masukan lebih lanjut. Adapun teori-teori yang berhubungan dengan penelitian tugas akhir ini.

2.2.1. *UCF Crime Dataset*



Gambar 2.1 UCF Crime Dataset

Pada penelitian ini, penulis menggunakan *UCF Crime Dataset*. *Dataset* yang dirancang oleh Sultani, dkk.[20], ini terbuat dari rekaman kamera pengawas yang dimana terdapat 13 jenis *anomaly* pada dunia nyata diantaranya Kekerasan, Penangkapan, Pembakaran, Penyerangan, Kecelakaan, Perampokan, Ledakan, Perkelahian, Pembegalan, Penembakan, Pencurian, Pengutilan, dan Vandalisme yang dimana contoh dari dataset ini dapat dilihat pada Gambar 2.1. Akan tetapi, pada penelitian ini penulis akan berfokus kepada *anomaly* Perampokan, Pembegalan, dan Pencurian dikarenakan kejahatan tersebut merupakan kejahatan paling banyak terjadi setiap tahun nya di indonesia berdasarkan data dari Badan Pusat Statistik Indonesia [21]. Dikarenakan ketiga jenis kejahatan tersebut melibatkan manusia, maka manusia akan dijadikan salah satu objek yang perlu diperhatikan oleh sistem yang akan penulis

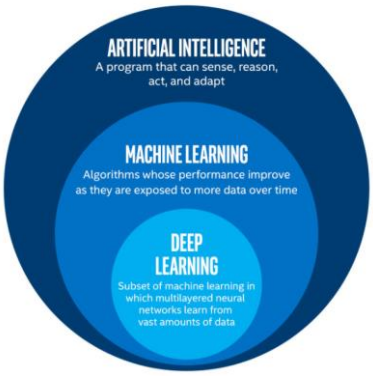
buat. Selain manusia, terdapat beberapa objek pada *dataset* ini yang akan dijadikan objek penting oleh penulis yaitu motor dan mobil [20].

2.2.2. *Closed Circuit Television* (CCTV)

Closed Circuit Television (CCTV) adalah alat perekam yang menggunakan satu atau lebih kamera *video* yang mengeluarkan *output* berupa *video* dan/atau audio [22]. Sistem keamanan berbasis CCTV merupakan sistem yang paling marak digunakan di banyak tempat di sekitar kita, baik itu rumah warga, toko, atau kantor pemerintahan. Sistem keamanan berbasis CCTV ini memberikan rasa aman terhadap orang-orang yang menggunakan tempat tersebut [23]. Namun tentunya sistem ini masih memiliki beberapa kekurangan, salah satunya adalah rekaman CCTV harus selalu dipantau oleh seorang *operator* secara real-time [24].

2.2.3. *Machine Learning*

Machine Learning adalah subset dari bidang *artificial intelligence*. *Machine Learning* adalah hasil perkawinan dari macam bidang, seperti *Artificial Intelligence, probability and statistics, computational complexity theory, control theory, information theory, philosophy, psychology, neurobiology* dan bidang-bidang lain [25]. Bidang ini berfokus untuk menjawab pertanyaan "bagaimana membuat program komputer yang bisa belajar dari pengalaman?" [25]. Bidang ini membuat program yang mempelajari suatu pola dari "masa lalu" yang dimana dalam hal ini adalah data yang telah dimiliki [26]. *Machine Learning* sudah digunakan di berbagai macam bidang ilmu pengetahuan mulai dari *Speech Recognition, Computer Vision, Bio-Surveillance, Robot Or Automation Control, Empirical Science Experiments* [26]. Bidang *Machine Learning* inilah yang nanti akan melahirkan bidang *Deep Learning.* Hubungan antara *Machine Learning, Deep Learning,* dan *Artificial Intelligence* dapat diamati pada Gambar 2.2.



Gambar 2.2 Hubungan Antara AI, ML, dan DL

2.2.4. *Deep Learning*

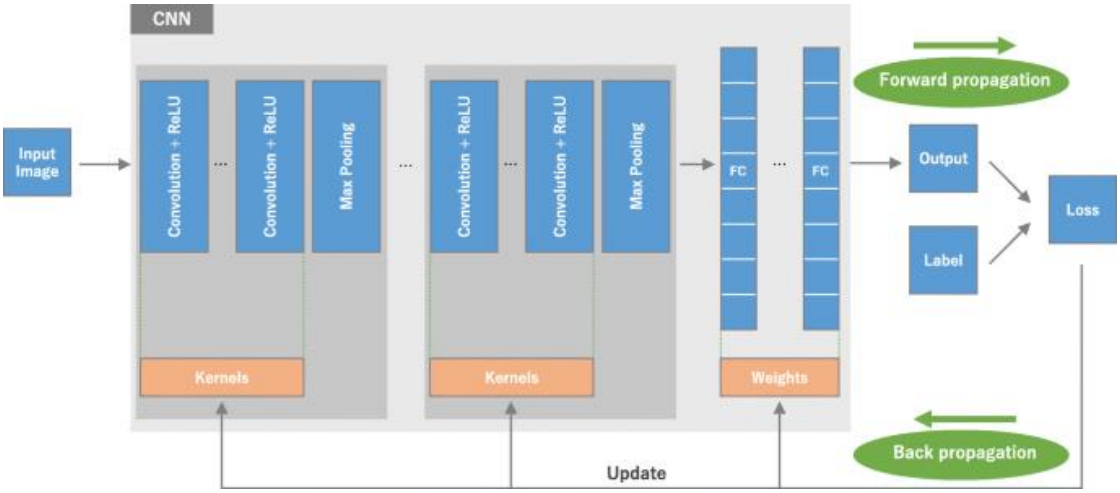
Deep Learning merupakan subset dari *Machine Learning* yang mengembangkan metode *Artificial Neural Network* (ANN). Metode *Artificial Neural Network* (ANN) ini diciptakan karena terinspirasi oleh bagaimana sistem *neuron* pada otak bekerja [27]. Bidang ini pada beberapa tahun belakangan telah mengalami perkembangan yang sangat pesat, hal ini dikarenakan semakin banyaknya jumlah data yang dapat digunakan untuk *training* model *Deep Learning*, serta dengan kemajuan teknologi *cloud computing* yang dapat mengurangi *computing power* yang diperlukan untuk melakukan training pada model *Deep Learning* membuat proses *training* lebih cepat [27]. Dikarenakan hal-hal tersebut, metode *Deep Learning* sudah marak digunakan untuk mengembangkan metode metode yang ada di bidang lain, seperti contohnya *Speech Recognition, Image Recognition dan Natural Language Processing* [27].

2.2.4.1. Deep Computer Vision

Deep Computer Vision adalah ilmu yang mempelajari cara mendapatkan informasi dari sebuah citra, seperti letak suatu objek, hal apa yang sedang terjadi pada citra tersebut untuk memprediksi dan mengantisipasi suatu kejadian di dunia menggunakan metode *Deep Learning* [28]. *Deep Computer Vision* adalah salah satu bidang yang mendapatkan dampak baik dengan berkembangnya metode *Deep Learning*, dimana mulai bermunculan neural network yang berbasis Convolutional Neural Network (CNN) setelah kesuksesan model CNN AlexNet yang muncul pada tahun 2012 [29]. Penggunaan *Deep Learning* pada bidang *Computer Vision* ini disebabkan karena adanya banyak sekali variasi fitur dari sebuah objek yang harus

diketahui, membuat pengembang kesulitan untuk mengekstraksi fitur yang baik secara manual [28]. Kemajuan ini membuat *Deep Computer Vision* memberi banyak dampak baik dibidang kesehatan, biologi, dan aksesibilitas [28].

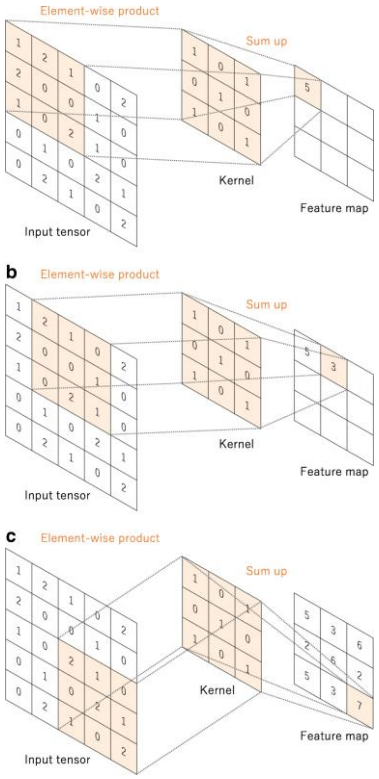
2.2.4.1. *Convolutional Neural Network* (CNN)



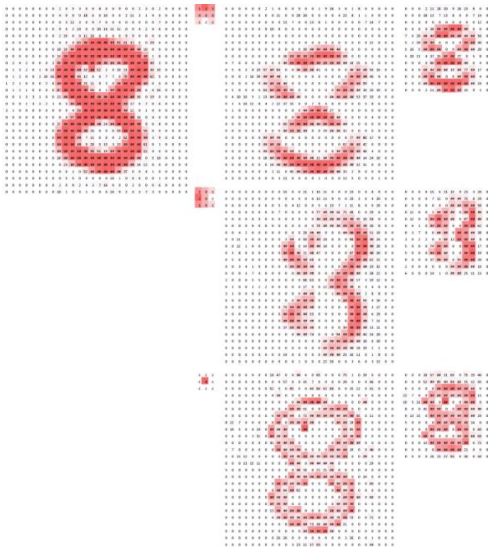
Gambar 2.3 Arsitektur CNN [30]

Arsitektur CNN dapat dilihat pada Gambar 2.3. Pada umumnya, CNN dapat dipecah menjadi beberapa bagian yaitu *input layer, convolutional layer, pooling layer, dan fully-connected layer* [31]. *Input layer* yang akan menerima dan menampung nilai pixel dari citra [31]. Kemudian ada *convolutional layer* yang akan menghitung output neuron pada layer tersebut berdasarkan nilai pada input serta layer-layer sebelumnya dengan menggunakan operasi *convolution* [30]. Operasi *Convolution* merupakan operasi yang digunakan untuk ekstraksi fitur, dimana sebuah *array 2D* yang disebut *kernel* dikalikan ke setiap *array input* yang disebut *tensor*. Adapun sub-matriks yang menjadi tempat suatu operasi perhitungan berjalan disebut *patch*. Hasil perhitungan *element-wise* dari *kernel* dan *tensor* dihitung pada setiap lokasi di *tensor* dan ditotalkan untuk mendapatkan nilai akhir pada setiap lokasi di *output tensor*

yang disebut *feature map* [30]. Adapun gambaran proses operasi *convolution* yang dapat diamati pada Gambar 2.4.



Gambar 2.4 Proses *Convolution* [30]



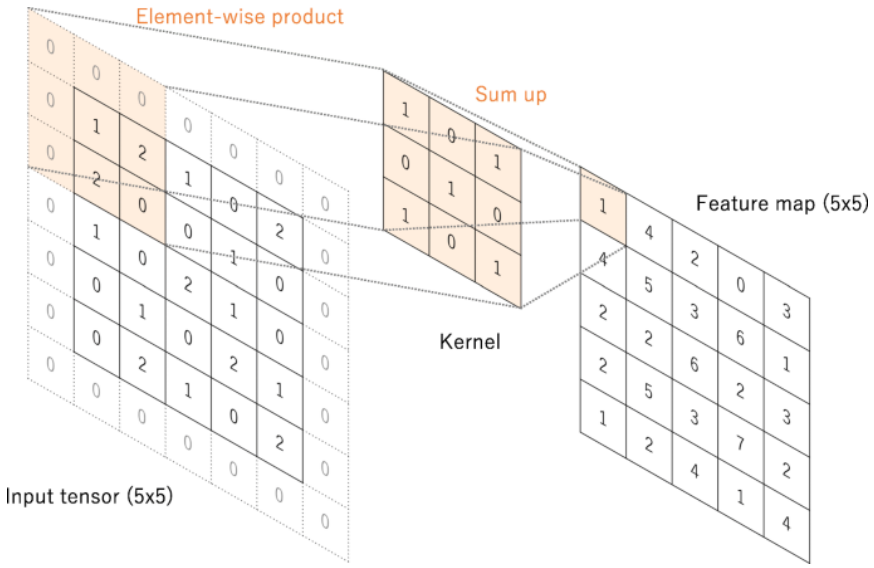
Gambar 2.5 Contoh cara kernel bekerja [30]

Gambar 2.5 menunjukan contoh bagaimana *kernel* bekerja. Gambar sebelah kanan adalah *input array* atau *tensor*, pada bagian tengah adalah *kernel*, serta

pada bagian kiri gambar adalah *feature map.* Dapat dilihat bahwa terdapat beberapa *kernel* dipekerjakan untuk mendapatkan fitur yang spesifik, seperti *horizontal edge detector* (atas), *vertical edge detector* (tengah), dan *outline detector* (bawah), kumpulan *kernel* ini disebut dengan *filter* [30]. Adapun rumus dari operasi *convolution* yang dapat diamati pada rumusan (2.1). Dimana nilai *m*, dan *n* merupakan lokasi *pixel* pada *input tensor*, *f* dan *g* merupakan *input tensor* dan *h* merupakan *kernel* serta *j* dan *k* merupakan lokasi nilai pada *kernel*.

$$G[m,n] = (f * g)[m,n] = \sum_j \sum_k h[j,k]\, f[m-j, n-k] \qquad (2.1)$$

Operasi *convolution* yang digunakan pada CNN biasanya tidak membiarkan bagian tengah dari *kernel* untuk diletakan di bagian paling luar *input tensor*, hal ini menyebabkan berkurangnya luas dari *feature map* yang dikeluarkan. Cara untuk menyelesaikan masalah ini adalah dengan menggunakan *Zero Padding*, dimana nilai nol ditambahkan pada setiap sisi *input tensor* membuat dimensi dari *tensor* terjaga [30]. Adapun gambaran *Zero Padding* yang dapat dilihat pada Gambar 2.6.



Gambar 2.6 *Padding* pada *Convolution Layer* [30]

Jarak antara kedua operasi kernel disebut *stride.* Pilihan nilai yang sering digunakan untuk *stride* adalah 1. Namun, nilai *stride* yang lebih besar biasanya digunakan untuk melakukan *downsampling* pada *feature map* [30]. Adapun metode lain untuk melakukan *downsampling* yaitu *Pooling* yang nanti akan penulis bahas. Tujuan proses *training* di CNN adalah untuk menemukan kernel terbaik berdasarkan *training dataset* [30]. *Kernel* adalah parameter didapatkan secara otomatis dari proses *training.* Di sisi lain, besar *kernel*, jumlah *kernel*, *padding*, and *stride* adalah *hyperparameters* yang harus di atur secara manual sebelum proses *training* dimulai [30]. Untuk detail setiap parameter dan hyperparameter dapat dilihat pada Tabel 2.2.

Tabel 2.2 *Parameter* dan *Hyperparameter* pada setiap layer [29]

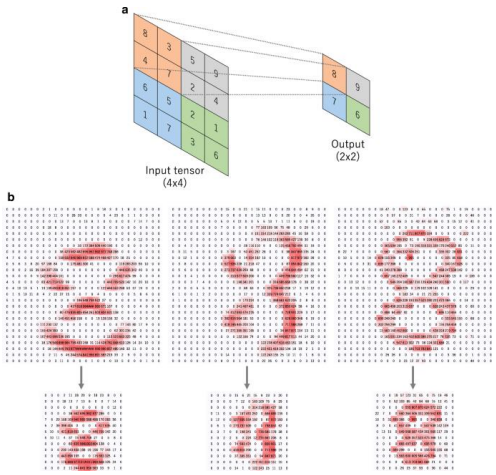
Layer	***Parameters***	***Hyperparmeters***
Convolution layer	*Kernels*	*Kernel size, number of kernels, stride, padding, activation function*
Pooling layer	*None*	*Pooling method, filter size, stride, padding*
Fully connected layer	*Weights*	*Number of weights, activation function*

Output dari operasi linear seperti *convolution* akan dilanjutkan ke dalam fungsi aktifasi. Fungsi Aktifasi yang biasa digunakan pada *output* proses *convolution* adalah ReLU. Adapun rumus ReLU diperlihatkan pada rumusan (2.2).

$$g(z) = max(0, z) \tag{2.2}$$

Setelah proses *convolution* dan juga aktfikasi *feature map* akan dimasukan ke *Pooling layer. Pooling Layer* akan melakukan *downsampling* pada *feature map* yang didapatkan. Proses *downsampling* digunakan untuk memperkenalkan *translation invariance* terhadap geseran dan distorsi serta memperkecil jumlah nilai yang perlu dihitung [30]. Metode *pooling* ada beberapa macam diantaranya *max* dan juga *average*. Pada saat *pooling* dilakukan, jika metode yang dilakukan menggunakan metode *max* maka *kernel* akan mencari nilai terbesar pada *kernel* untuk setiap lokasi di *feature map* untuk dijadikan nilai

pada *output tensor*. Jika metode yang digunakan adalah metode *average* maka kernel akan mencari nilai rata-rata pada *kernel* untuk dijadikan nilai pada *output tensor*. Adapun proses *pooling* serta contoh proses pooling yang dapat dilihat pada Gambar 2.7 serta rumus metode *max* dan *average* pada layer *pooling* yang dapat diamati pada rumusan (2.3) dan (2.4). Untuk rumusan (2.3), *X* adalah *patch* dari *feature maps* yang dijadikan *input* pada *pooling layer*. Lalu pada rumusan (2.4), p dan q adalah posisi ujung atas kiri dari *filter pooling*, m dan n adalah besar dari *filter pooling*, serta i dan j adalah *index pixel* pada *input tensor*.



Gambar 2.7 **a.** Proses *Pooling* dengan metode max; **b.** Contoh hasil Proses *Pooling* [30]

$$x = \max(X) \tag{2.3}$$

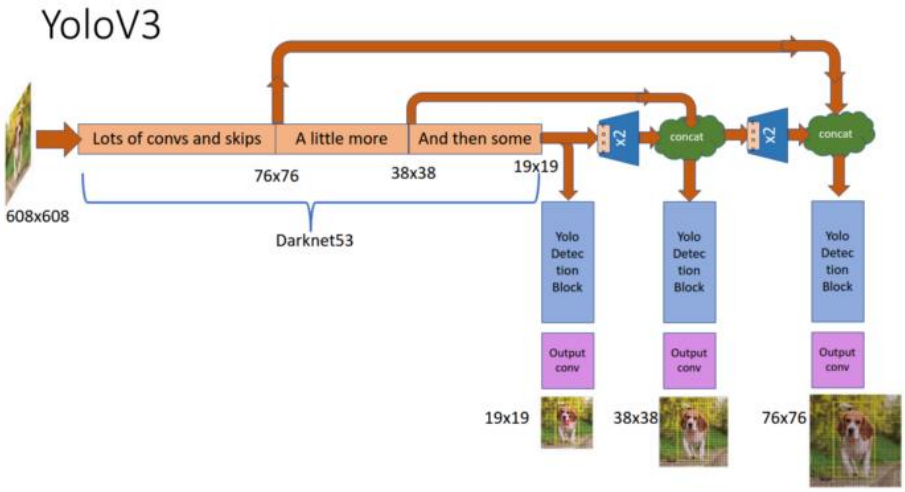
$$x = \frac{\sum_{i=p}^{p+m-1} \sum_{j=q}^{q+n-1} X_{i,j}}{m+n} \tag{2.4}$$

Fully-connected layer akan melakukan pekerjaan yang sama dengan ANN pada umumnya [31]. *Output* pada fase pertama (*convolution* dan *pooling* secara berulang-ulang) diratakan menjadi satu dimensi dan dimasukan ke dalam *fully-connected layer* yang nantinya akan menghasilkan nilai akhir berupa probabilitas untuk setiap kelas yang ada pada sebuah kasus klasifikasi. Setiap *layer* pada *fully-connected layer* diikuti dengan fungsi aktifasi seperti ReLU, Sigmoid, dan Tanh [30]. Pada *layer* terakhir di *fully-connected layer* fungsi aktifasi yang digunakan biasanya berbeda dengan fungsi aktifasi yang

digunakan di *layer* lain [30]. Pada layer ini fungsi aktifasi yang sering digunakan adalah fungsi *softmax* yang menormalisasikan setiap *output* dari nilai asli ke nilai antara 0 dan 1 [30]. Adapun rumus *soft*max yang dapat dilihat pada rumusan (2.5) dimana $\vec{z}$ merupakan vector input.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.5}$$

2.2.4.2. *You Only Look Once Version 3* (YOLOv3)



Gambar 2.8 Arsitektur YOLOv3 [32]

You Only Look Once Version 3 (YOLOv3) merupakan algoritma yang digunakan untuk mendeteksi objek pada suatu citra. Gambar 2.8 menunjukan arsitektur dari YOLOv3 dimana algoritma menerima *input* dengan panjang dan lebar yang dapat dibagi oleh angka 32. Hal ini dilakukan karena ketiga *detection layer* yang dimiliki oleh algoritma ini mempunyai nilai *stride* sebesar 32, 16, 8 [33] pembahasan *detection layer* akan penulis lanjutkan nanti. Setelah citra dalam bentuk *tensor* dimasukan ke dalam algoritma, citra akan diekstrasi fiturnya melalui darknet-53 yang dimana arstektur darknet-53 dapat diliat pada Gambar 2.9. Setelah dilakukan tahap ekstraksi, *feature map* yang didapat akan masukan ke dalam tiga *detection layer*.

Ketiga Detection layer ini memiliki fungsi yang berbeda dengan ukuran input yang berbeda juga. *Detection layer* yang pertama berfungsi untuk mendeteksi

objek besar, *detection layer* kedua berfungsi untuk mendeteksi objek sedang dan *detection layer* ketiga berfungsi untuk mendeteksi objek kecil [33]. Untuk menghubungkan ketiga *detection layer*, *upsampling layer* ditambahkan. *Layer* ini digunakan untuk memperbesar ukuran *feature map* yang didapat setelah dilakukan *downsampling* terus menerus pada tahap ekstraksi fitur menggunakan darknet-53.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8

Gambar 2.9 Arsitektur Darknet-53 [18].

Secara garis besar, algoritma YOLO memprediksi sebuah objek pada citra dilakukan dengan beberapa tahap. Citra yang akan diprediksi akan dibagi ke dalam beberapa *grid cell*. Jumlah *grid cell* ditentukan dengan membagi banyaknya piksel pada citra input dengan nilai *stride*. Jika titik tengah objek jatuh pada suatu grid, maka grid tersebut bertanggung jawab untuk mendeteksi objek tersebut [14].

Setiap *grid cell* akan memprediksi *bounding box* dan *confidence score* untuk *bounding box* tersebut. *Confidence score* ini nantinya akan merepresentasikan seberapa yakin algoritma bahwa *bounding box* tersebut berisi objek dan seberapa akurat *bounding box* tersebut [14]. Rumus *confidence score* dapat

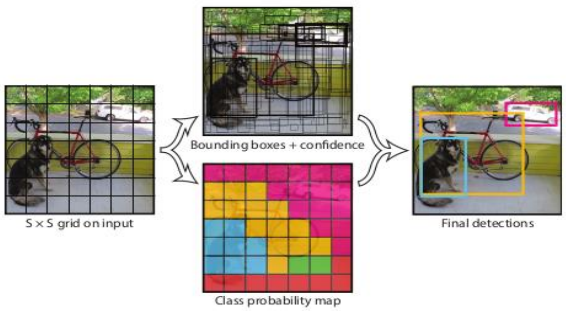
dilihat pada rumusan (2.7) [14]. Jika grid cell tidak memiliki objek di dalamnya, maka nilai $\Pr(\text{Object}) = 0$ dan jika grid cell memiliki objek di dalamnya, maka nilai $\Pr(\text{Object}) = 1$ [14]. Serta pada rumusan (2.7) terdapat juga $IOU_{truth\ pred}$ yang dimana adalah *Intersection Over Union* yang merepresentasikan seberapa baik prediksi *bounding box*. Setiap *Bounding box* memiliki 5 nilai yang akan diprediksi yaitu *x, y, w, h,* dan *confidence* [14]. Nilai (x, y) merepresentasikan koordinat tengah dari *bounding box* relatif ke batasan setiap *grid cell* [14].

Setiap *grid cell* juga memprediksi C sebagai nilai *conditional class probabilities*, yang rumusnya dapat dilihat pada rumusan (2.6) [14]. Probabilitas ini berada pada grid cell yang memiliki objek [14]. Model hanya akan memprediksi satu *set* probabilitas kelas untuk setiap *grid cell* tanpa memperdulikan jumlah *bounding box* pada grid tersebut. Pada tahap test, model melakukan perkalian antara *conditional class probabilities* dan *confidence* seperti pada rumus (2.8) yang akan memberikan *confidence score* pada setiap *class* di setiap *box*. Score ini memiliki nilai yang representasikan probabilitas *class* dan seberapa baik hasil *bounding box* yang diprediksi dalam mengalokasikan objek [14]. Adapun gambaran alur algoritma YOLO dapat dilihat pada Gambar 2.10.

$$\mathrm{C} = \Pr(Class_i | \ Object) \tag{2.6}$$

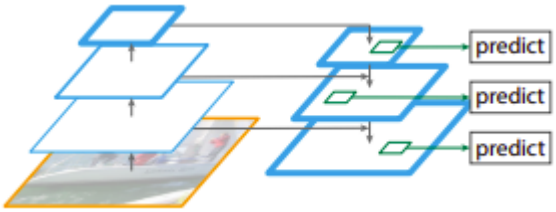
$$confidence = \Pr(\text{Object}) \ * \ IOU_{truth\ pred} \tag{2.7}$$

$$\begin{aligned} &\Pr\left(Class_i \mid Object\right) \ * \ \Pr\left(Object\right) \ * \ IOU_{truth\ pred} \\ &= \ Pr(Class_i) \ * IOU_{truth\ pred} \end{aligned} \tag{2.8}$$



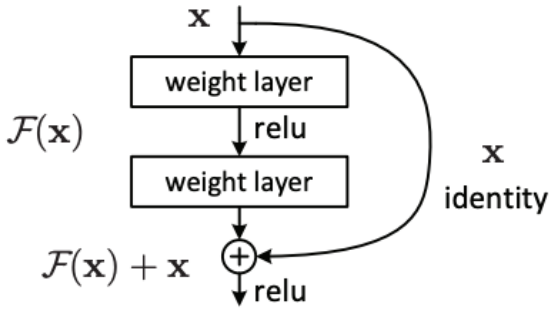
Gambar 2.10 Alur Deteksi Algoritma YOLO [14]

YOLOv3 sendiri menambahkan beberapa metode baru pada arsitekturnya, yaitu menambahkan sistem yang melakukan prediksi pada 3 ukuran yang berbeda. Cara ini mirip dengan penerapan algoritma *Feature Pyramid Network* (FPN) [34] yang dapat dilihat pada Gambar 2.11.



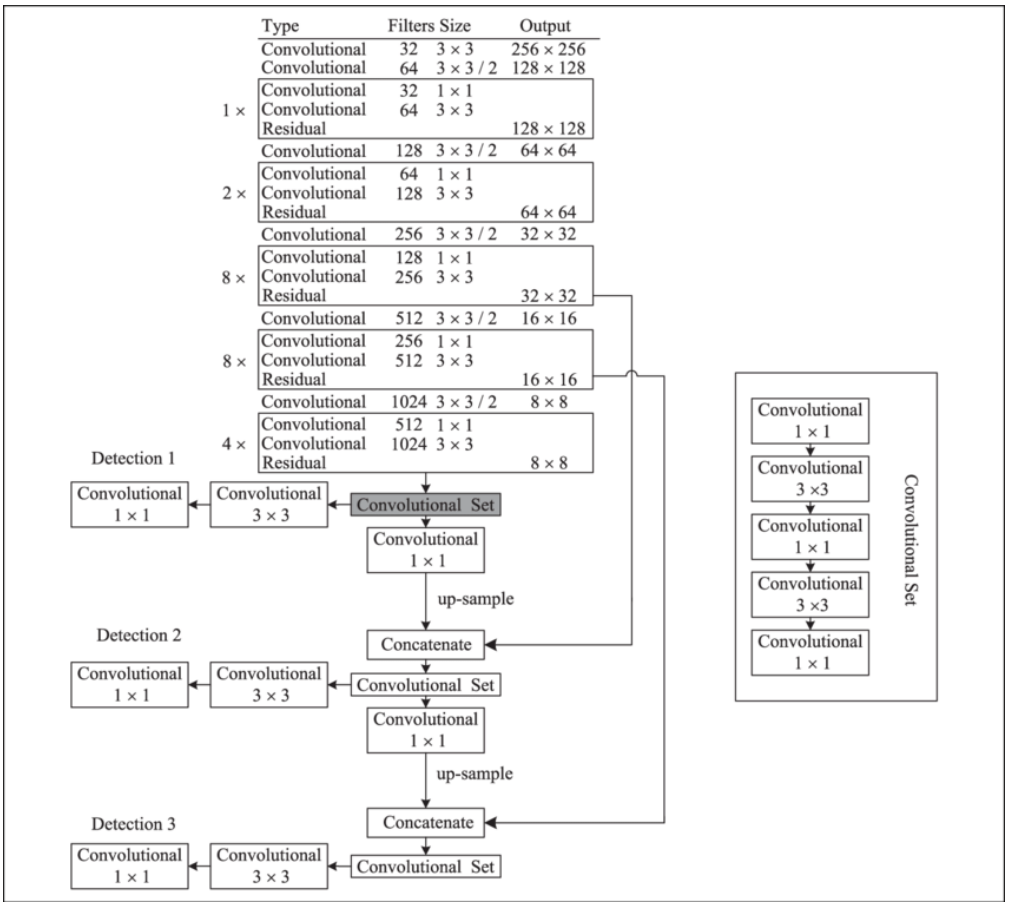
Gambar 2.11 Feature Pyramid Network *[34]*.

Pada Gambar 2.9 terdapat satu komponen yang belum penulis bahas yaitu *residual block. Residual Block* adalah blok pada suatu algoritma *Deep Learning* yang melakukan "*shortcut connections"*. *Shortcut connections* terjadi jika ada *layer* pada algoritma *deep learning* yang mengeluarkan output dengan menghiraukan satu atau lebih layer setelahnya seperti pada Gambar 2.12 [35]. Metode ini dapat menyelesaikan masalah degradasi pada akurasi *training* yang dialami algortima *Deep Learning* dengan jumlah *layer* yang tinggi [35].



Gambar 2.12 Residual Block [35]

Pada proses di dalam *residual block* ini, sebelum *feature maps* diberikan ke *layer* selanjutnya *feature maps* yang murni disimpan yang nantinya akan digunakan untuk melakukan *identity mapping* dengan rumusan (2.9) diakhir *residual block*. Semua metode di atas digabungkan untuk menjadi algoritma YOLOv3 yang rinciannya dapat dilihat pada Gambar 2.13.



Gambar 2.13 Rincian Arsitektur YOLOv3 [36].

$$y = F(x, \{W_i\}) + x \tag{2.9}$$

Dimana:
$F(x, \{W_i\})$ = Hasil perhitungan weight layer
x = Identity map.

2.2.5. *Optical Flow Estimation*

Optical flow estimation merupakan teknik yang sering digunakan pada permasalahan pendeteksian gerak untuk mendapatkan informasi berupa vektor kecepatan [12]. Sistem bekerja dengan cara menghitung nilai kecerahan pada dua frame yang berdekatan, jika dua frame tersebut diambil pada waktu yang berdekatan maka kemungkinan nilai kecerahan yang dimiliki pada suatu pixel tidak banyak berubah sehingga dapat dapat dihitung vektor kecepatannya berdasarkan kedua frame tersebut.

2.2.5.1. FastFlowNet

FastFlowNet adalah model *Deep Learning* pada bidang *Computer Vision* yang didesain untuk memproses gambar dan mengeluarkan prediksi berupa Estimasi *Optical Flow* dari sebuah video dengan performa yang cepat dan akurasi yang tinggi [15]. Pada penelitian yang ditulis oleh Ka Man Lo disebutkan bahwa kecepatan FastFlowNet lebih cepat 10x lipat dibandingkan model RAFT [12]. FastFlowNet menggunakan model FlowNet [37] sebagai dasar arsitekturnya.

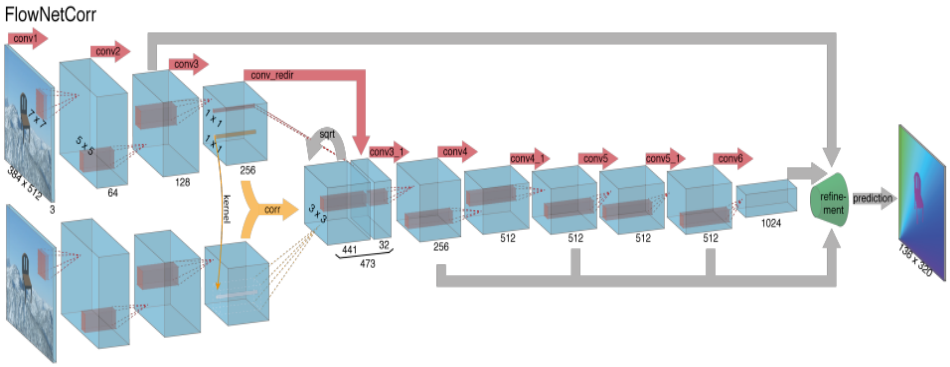
FlowNet merupakan model berbasis CNN yang menerima dua citra yang nanti akan diproses melalui dua alur proses yang terpisah namun memiliki arsitektur yang sama, setelah itu hasil dari kedua proses itu akan disatukan ke dalam arus yang sama dengan memanfaatkan *correlation layer* dimana proses *correlation* memiliki perhitungan seperti yang dapat diamati pada rumusan (2.10).

$$c(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle \tag{2.10}$$

$f_1 = feature\ maps\ pertama$
$f_2 = feature\ maps\ kedua$
$x_1 = nilai\ vector\ posisi\ pixel\ pada\ f_1$

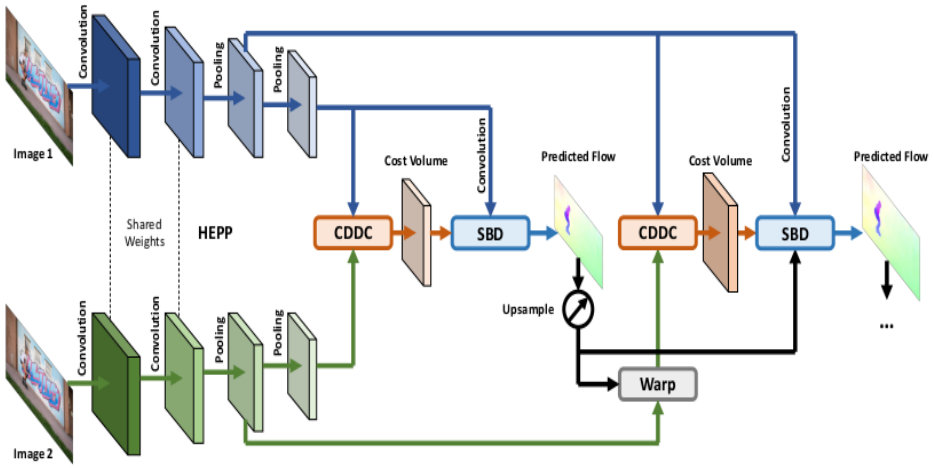
$$x_2 = nilai\ vector\ posisi\ pixel\ pada\ f_2$$

Dimana $f_1$dan f_2 merupakan *feature maps*, $x_1 dan\, x_2$ adalah *vector* yang menunjukan posisi tengah dari *patch* pada kedua *feature maps* dan k adalah besar *patch* [37]. Setelah proses correlation dan tahap convolution selesai model ini akan melakukan tahap refinement yang menambahkan proses *upconvolutional* layer yang terdiri dari *up pooling layer* dan *convolutional layer* [37]. Adapun gambaran arsitektur dari algoritma FlowNet dapat diamati pada Gambar 2.14.



Gambar 2.14 Arsitektur FlowNet [37]

Berdasarkan model FlowNet yang dijelaskan diatas,.FastFlowNet membuat beberapa perubahan yaitu dengan menambahkan atau mengganti beberapa metode yaitu dengan menambahkan *Head Enhanced Pooling Pyramid (HEPP)* yang terinspirasi dari model PWC-Net [38], *Center Dense Dilated Correlation (CDDC)*, *Shuffle Block Decoder (SBD)* yang diajukan oleh peneliti dengan memanfaatkan keuntungan yang didapat dari keluaran CDDC [15]. Yang mana secara garis besar arsitektur dari algoritma FlowNet dapat diamati pada Gambar 2.15.



Gambar 2.15 Arsitektur FastFlowNet [15].

Sama dengan versi sebelumnya, FastFlowNet menerima dua citra berbeda yang masing-masing akan dimasukan kedalam HEPP yang terdiri dari *convolution layer* dan *pooling layer* [15]. Beberapa *tensor output* dari *pooling layer* pada tahap HEPP akan digunakan kembali pada saat estimasi. Serta hasil akhir dari kedua proses ini akan dimasukan ke dalam proses estimasi. FastFlowNet melakukan lima kali estimasi di *scale* citra yang berbeda-beda. Dimana setiap proses estimasi terdapat 2 tahapan yaitu CDDC dan juga SBD.

Pada proses estimasi pertama setelah proses HEPP *model* menggunakan kedua *feature maps* dan memasukan ke dua *feature maps* tersebut ke dalam CDDC untuk menghitung *cost volume* [15]. *Cost volume* adalah hasil perhitungan korelasi antara dua *feature map* yang didapatkan dari proses sebelumnya dengan menggunakan rumus yang dapat dilihat pada rumusan (2.11). *Cost Volume* digabungkan dengan hasil satu *layer pooling* terakhir pada tahap HEPP untuk dilakukan proses *decoding* menggunakan metode SBD untuk mendapatkan estimasi *optical flow* [15]. Sebelum melakukan proses *decoding*, algoritma melakukan *group convolution* yaitu proses mengelompokan parameter yang sejenis. Pada kasus ini, kelompok parameter adalah hasil estimasi *optical flow* dari tahap sebelumnya dimana untuk proses estimasi yang pertama merupakan nilai ini merupakan matriks nol, *Cost Volume* dari CDDC, serta hasil *pooling* dari salah satu *pooling layer* pada proses HEPP. Setelah itu SBD melakukan *channel shuffle operations* yang didapat dari penelitian yang

buat oleh Zhang, dkk. [39]. Lalu dilakukan proses *decoding* untuk mendapatkan hasil estimasi *optical flow*.

Pada proses estimasi selanjutnya yang hampir mirip dengan proses estimasi pertama. Hanya saja, pada proses CDDC ini salah satu *feature map* dari proses HEPP dilakukan operasi *wraping* berdasarkan hasil estimasi *optical flow* dari proses sebelumnya yang telah dilakukan 2x *upsampling* lalu itu dihitung *cost volume*nya dengan menggunakan rumus (2.11) [15]. Setelah didapatkan *cost volume,* proses dilanjutkan ke tahap SBD. Dimana pada proses estimasi ini algoritma menggabungkan *cost volume* yang didapatkan dari proses CDDC dengan estimasi *optical flow* dari tahap sebelumnya yang telah dilakukan 2x *upsampling* serta *feature map* dari salah satu proses *pooling* di tahap HEPP. Lalu SBD akan melakukan *decoding* yang sama seperti pada proses estimasi pertama untuk mendapatkan estimasi *optical flow* [15]. Adapun rincian arsitektur algoritma FastFlowNet dapat dilihat pada Gambar 2.16 dimana block pertama mendeskripsikan tahap HEPP, block kedua mendeskripsikan tahap CDDC 1 sampai 5 dan block terakhir menggambarkan tahap SBD 1 sampai 5.

Layer Name	Kernel	Stride	Input
pconv1_1	3×3	2	img1 or img2
pconv1_2	3×3	1	pconv1_1
pconv2_1	3×3	2	pconv1_2
pconv2_2	3×3	1	pconv2_1
pconv2_3	3×3	1	pconv2_2
pconv3_1	3×3	2	pconv2_3
pconv3_2	3×3	1	pconv3_1
pconv3_3	3×3	1	pconv3_2
pool4	2×2	2	pconv3_3
pool5	2×2	2	pool4
pool6	2×2	2	pool5
upconv6	4×4	1/2	flow6
warp5	-	-	pool5b,upconv6
cddc5	1×1	1	pool5a,warp5
rconv5	3×3	1	pool5a
concat5	rconv5 + cddc5 + upconv6		
fconv5_1	3×3	1	concat5
fconv5_2	3×3	1	fconv5_1
shuffle5_2	-	-	fconv5_2
fconv5_3	3×3	1	shuffle5_2
shuffle5_3	-	-	fconv5_3
fconv5_4	3×3	1	shuffle5_3
shuffle5_4	-	-	fconv5_4
fconv5_5	3×3	1	shuffle5_4
fconv5_6	3×3	1	fconv5_5
fconv5_7	3×3	1	fconv5_6

Gambar 2.16 Rincian Arsitektur FastFlowNet [15]

$$c^{\cdot}(x,d) = \frac{f_1^l(x) \,.\, f_{warp}^l(x+d)}{N}, d \in [-r,r] \times [-r,r] \quad (2.11)$$

x = vector 2 dimensi berisi lokasi pixel
d = radius dari perhitungan
$f_1^l = fitur\ map\ pertama$
$f_{warp}^l = fitur\ map\ kedua\ yang\ sudah\ dilakukan\ wraping\ dengan\ estimasi\ pada\ proses\ sebelumnya\ jika\ ada. Jika\ tidak\ maka\ fitur\ map\ kedua\ saja\ tanpa\ perlu\ diwraping$

2.2.6. *Tools*

Pada bagian ini penulis akan membahas tentang *tools* yang akan digunakan penulis saat penelitian ini berlangsung

2.2.6.1. Pytorch

Pytorch adalah machine learning framework yang populer, biasa digunakan untuk aplikasi Computer Vision dan juga Natural Language Processing. Machine Learning library yang dikembangkan oleh *Meta* ini berfokus pada kemudahan penggunaan dan juga kecepatan [40]. Library ini mempermudah programmer untuk menulis kode serta melakukan debugging pada model yang telah dibuatnya [40].

2.2.6.2. Python

Python adalah bahasa dinamis *high-level* yang menawarkan kemudahan untuk mempelajarinya dikarenakan *syntax* yang dimiliki bahasa pemrograman tersebut lebih menyerupai bahasa manusia. Desain dari bahasa ini berfokus pada keterbacaan, dan membuat *syntax* yang mempermudah *programmer* mengekspresikan sebuah konsep dengan baris kode yang lebih sedikit [41]. Bahasa ini pula dapat digabungkan dengan mudah dengan bahasa C/C++ membuat aplikasi yang di bangun memiliki performa yang tinggi [41].

2.2.6.3. OpenCV

OpenCV adalah merupakan *library* yang dibuat bertujuan untuk mempermudah pengolahan citra [42]. *Library* ini dapat diimplementasi menggunakan bahasa C++, Python, Java, dan MATLAB.

2.2.7. *Confusion Matrix*

Confusion Matrix adalah alat untuk menganalisa hasil prediksi dari sebuah model Machine Learning [43]. Matrix ini terdiri dari 4 nilai, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). *True Positive* (TP) adalah nilai dimana nilai sebenarnya adalah positif dan juga prediksi model juga positif, *False Positive* (FP) adalah saat nilai sebenarnya negatif sedangkan prediksi dari model adalah positif, *False Negative* (FN) adalah saat nilai sebenarnya negatif dan

prediksi model negatif, dan *True Negative* (TN) adalah saat nilai sebenarnya positif tetapi nilai prediksinya negatif[25]. Adapun gambaran tentang *confusion matrix* yang dapat dilihat pada Tabel 2.3.

Tabel 2.3 *Confusion matrix*

		Nilai Sebenarnya	
		Positif	**Negatif**
Nilai Prediksi Model	**Positif**	TP	FP
	Negatif	FN	TN

2.2.8. *Precision*

Precision adalah salah satu metode validasi model *Machine Learning*, Nilai ini akan menentukan keandalan model *Machine Learning* yang divalidasi [43]. Adapun rumus *precision* yang dapat diamati pada rumusan (2.12).

$$Precision = \frac{TP}{TP + TN} \tag{2.12}$$

2.2.9. *Recall*

Recall adalah salah satu metode validasi model *Machine Learning*. Nilai ini mengukur seberapa banyak nilai positif yang dapat diprediksi dengan benar oleh model, Nilai recall yang tinggi artinya nilai diprediksi dengan benar (TP + FN) kebanyakan akan dilabeli sebagai (TP), hal ini dapat menimbulkan nilai FP yang besar tetapi akurasi yang kecil [43]. Sebaliknya jika nilai recall kecil maka nilai TP + FN, maka nilai FN akan cenderung lebih banyak yang artinya bahwa nilai yang dilabel TP oleh model akan lebih meyakinkan [43]. Adapun rumus *recall* dapat diamati pada rumusan (2.13).

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

2.2.10. *F1-Score*

Ketika merancang sebuah model machine learning saat kita ingin meningkatkan precision dari model tersebut maka nilai recall akan turun begitu pula sebaliknya. F1-Score merupakan nilai rata-rata harmonis antara recall dan juga precision. Dengan menggunakan nilai *F1-Score* maka akan didapatkan gambaran tentang nilai *recall* dan

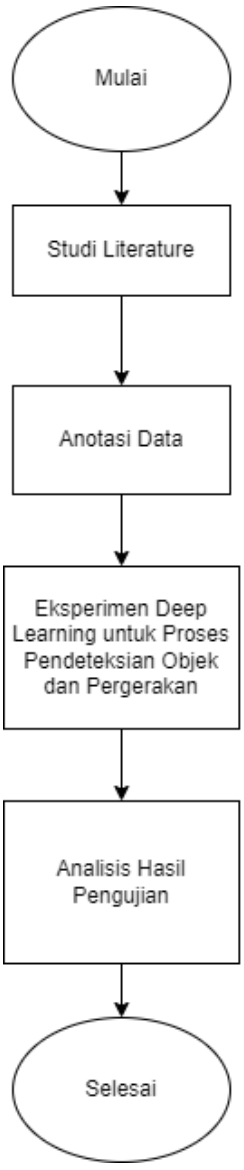
juga *precision* [43]. Nilai tertinggi akan didapat jika nilai *recall* dan juga *precision* sama. Adapun rumus F1-Score yang dapat dilihat pada rumusan (2.14).

$$F1 = \frac{1}{\frac{1}{recall} + \frac{1}{precision}} \tag{2.14}$$

BAB III
METODE PENELITIAN

3.1.Alur Penelitian

Untuk memperjelas penelitian yang akan dilakukan oleh penulis maka dibuatlah alur penelitian yang terdiri dari tahap-tahap diantaranya Studi Literatur, Akusisi Data, Anotasi Data, Eksperimen *Deep Learning* untuk Proses Pendeteksian Objek den Pergerakan, Pengujian Model, serta Analisis Hasil Pengujian. Adapun alur penelitian yang digambarkan di dalam bentuk diagram alir yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

3.2.Penjabaran Langkah Penelitian

Pada bagian ini penulis akan menjabarkan setiap Langkah penelitian yang ditampilkan pada Gambar 3.1.

3.2.1. Studi Lituratur

Pada tahapan ini penulis melakukan pengayaan pengetahuan melalui jurnal-jurnal internasional maupun nasional untuk membandingkan serta mendapatkan gambar tentang penelitian yang akan penulis dilakukan. Diharapkan dengan selesainya tahapan studi literatur penulis akan memiliki pemahaman yang lebih dalam saat memulai penelitian dan juga dapat menyelesaikan segala permasalahan yang muncul saat melakukan penelitian.

3.2.2. Anotasi Data

Pada bagian ini penulis akan menjelaskan tahapan anotasi data. Pada tahapan ini penulis akan melakukan anotasi pada setiap *frame UCF Crime Dataset*. Dimana anotasi yang dilakukan akan melabel bagian mana pada *video* yang terdapat pergerakan objek penting, sedangkan bagian *video* yang tidak penting tidak akan dilabel. Adapun bentukan anotasi yang akan dilakukan dapat dilihat pada Gambar 3.2.

<Nama File>	<Frame awal>	<Frame Akhir>	<Penting?>
Arrest001_x264.	1185	1485	Penting

Gambar 3.2 Anotasi Data

3.2.3. Eksperimen Deep Learning untuk Proses Deteksi Objek dan Gerak

Pada tahap ini penulis akan membahas komponen yang diperlukan untuk melakukan eksperimen penelitian seperti alat, bahan, arsitektur sistem, dan tahapan eksperimen.

3.2.4.1. Alat

Pada penelitian ini penulis menggunakan beberapa alat untuk menyelesaikan penelitiannya, alat-alat tersebut diantaranya adalah:

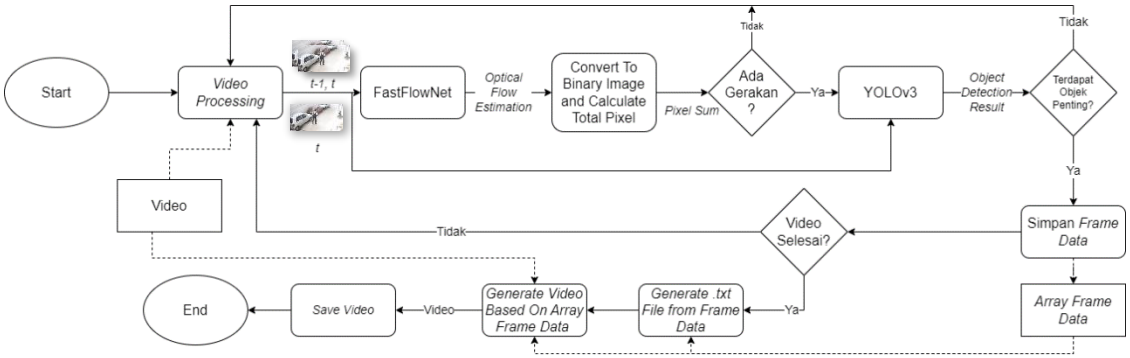
1. Laptop dengan spesifikasi Intel® Core™ i3-5005U CPU @ 2.00GHz × 4, GPU NVIDIA Corporation GM108M [GeForce 930M] / Mesa Intel® HD Graphics 5500 (BDW GT2), HDD space 1,0 TB, Memory 8GB DDR3.
2. *Google Collab* untuk *Development Environment* Model *Deep Learning*
3. *Visual Studio Code* untuk *Development Environment* saat membangun aplikasi

3.2.4.2. Bahan

Pada penelitian ini dibutuhkan bahan yang dibutuhkan dalam penelitian untuk menunjang pengembangan aplikasi. Bahan-bahan yang dibutuhkan yaitu :

1. UCF Crime Dataset yang berfokus pada anomali Perampokan, Pembegalan, dan Pencurian.

3.2.4.3. Pengembangan Arsitektur



Gambar 3.3 Arsitektur sistem yang dikembangkan

Pada penelitian ini penulis mengembangkan arsitektur sistem untuk melakukan segmentasi *video* rekaman kamera pengawas dengan menggunakan algoritma FastFlowNet dan juga YOLOv3. Dimana rancangan sistem dapat dilihat pada Gambar 3.3. Sistem dimulai dengan memasukan *video* dengan resolusi 320 x 240 dan *frame*

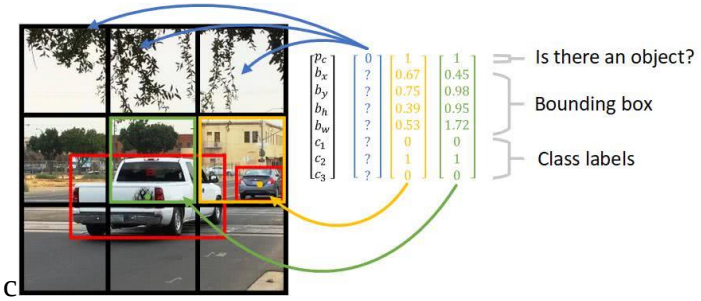
rate 30fps ke dalam sistem yang mana *video* ini akan diproses pada tahap *video processing* untuk diambil *frame*nya lalu dilakukan *resize* ke ukuran 320 x 320.

Gambar 3.4 Contoh Keluaran FastFlowNet (2 gambar pertama: input; gambar terakhir: output;)

Sistem akan memulai mengambil gambar dari frame ke 2 yang dimana hal ini dikarenakan dibutuhkan dua gambar untuk dimasukan ke dalam FastFlowNet (t dan t-1). Model FastFlowNet yang akan digunakan penulis pada penelitian ini merupakan model *pre-trained.* Dimana model ini memiliki arsitektur yang sama seperti pada pembahasan di sub bab 2.2.5.1. Setelah didapatkan *frame* ke t dan *frame* ke t-1 (t adalah waktu pada video), sistem memasukan kedua *frame* tersebut ke dalam algoritma FastFlowNet yang akan mengeluarkan hasil berupa *optical flow estimation*. Adapun contoh hasil *optical flow estimation* yang dikeluarkan oleh fastflownet yang dapat dilihat pada Gambar 3.4. Lalu hasil *optical flow estimation* ini akan diubah ke dalam bentuk binary image dan dihitung total *pixel*nya. Nilai total *pixel* ini akan dibandingkan dengan *threshold* yang ditentukan oleh penulis untuk menentukan apakah proses dapat dilanjutkan ke pendeteksian objek atau tidak. Jika nilai total *pixel* tidak melebihi *threshold* maka proses akan kembali ke tahap awal yaitu *video processing*. Jika nilai total pixel melebihi *threshold* maka proses akan dilanjutkan ke dalam proses pendeteksian objek dengan algoritma YOLOv3. Adapun rentan nilai *threshold* pada penelitian ini yaitu dari nilai 1000 sampai 10000 yang didapatkan dari penelitian Gozali dan Akbar [44] serta penelitian Thohari dan Ramadhani [45].

Pada tahap pendeteksian objek, model YOLOv3 yang digunakan adalah model *pre-trained* yang memiliki arsitektur yang sama seperti pada pembahasan di sub bab 2.2.4.2. Algoritma YOLOv3 akan mendapatkan masukan dari bagian *video processing* berupa *frame* ke t pada *video* yang telah dilakukan *resize* ke ukuran 320 x 320. Setelah didapatkan *frame t* maka YOLOv3 akan melakukan proses pendeteksian objek yang menghasilkan matriks *output*. *Output* YOLOv3 berupa matriks berisi letak *bounding*

box, *confidential score*, serta label-label *class* yang diklasifikasikan pada setiap *grid* [14]. Adapun contoh luaran YOLOv3 yang dapat dilihat pada Gambar 3.5.



Gambar 3.5 Contoh output YOLOv3 (Matriks dengan besar 3 x 3 x 8)

Sistem akan memanfaatkan *output* ini untuk mendeteksi objek dengan label manusia, mobil atau motor. Jika objek yang terdeteksi adalah objek penting seperti manusia, mobil, atau motor. Maka sistem akan mengklasifikasikan bahwa terdapat pergerakan objek penting pada *frame* ke-t yang nanti hasil klasifikasi ini akan disimpan dalam objek dengan nilai Nama *File*, *Frame* ke, dan *label* apakah ada pergerakan objek penting atau tidak. Adapun gambaran objek *Frame Data* yang dapat dilihat pada Gambar 3.6. Objek ini akan disimpan ke dalam *Array* sampai proses selesai. Proses ini akan tersebut berulang sampai sistem kehabisan *frame* untuk diproses.

Nama File	Frame Ke	Pergerakan Objek Penting ?
Robbery_1.mp4	200	Ya

Gambar 3.6 Contoh *Frame Data*

Setelah proses pendeteksian pergerakan dan pendeteksian objek selesai, maka sistem akan membuat serta menyimpan data dari *Array Frame Data* yang telah dibuat di tahap sebelumnya ke dalam format .txt dan dalam bentuk yang sama seperti pada Gambar 3.2. File .txt ini nantinya akan digunakan untuk melakukan pengecekan performa dari sistem. Lalu *Frame Data* yang telah dikumpulkan pada proses sebelumnya akan digunakan untuk melakukan segmentasi *video* berdasarkan *Frame Data* yang telah didapatkan. *Video* akan disegmentasi dengan cara membuang bagian yang tidak

memiliki pergerakan manusia atau objek penting. Setelah itu *video* akan disimpan ke *hard drive.*

3.2.4.4. Tahap Eksperimen

Pada tahap ini penulis melakukan eksperimen pendeteksian objek dan juga pendeteksian gerak menggunakan algoritma YOLOv3 dan juga FastFlowNet. Adapun tahapan eksperimen yang akan dilakukan oleh penulis dalam penelitan ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Tahap Eksperimen

Eksperimen Ke-	Eksperimen yang dilakukan	Luaran Eksperimen
1.	Melakukan eksperimen untuk mencari nilai *threshold* yang tepat untuk menentukan adanya pergerakan atau tidak dari luaran algoritma FastFlowNet	Diketahui nilai *threshold* yang paling efektif dalam pendeteksian pergerakan objek penting dari luaran algoritma FastFlowNet.
2.	Melakukan eksperimen penggabungan klasifikasi gerakan dan juga deteksi objek algoritma FastFlowNet dan juga YOLOv3.	Menghasilkan algoritma gabungan dari FastFlowNet dan YOLOv3 dengan input berupa video dan output berapa label ada pergerakan penting atau tidak pada setiap frame
3.	Pengujian hasil penggabungan algoritma FastFlowNet dan juga YOLOv3.	Hasil perhitungan F1-Score berdasarkan nilai label ground truth dan label prediksi
4.	Melakukan eskperimen untuk melakukan pemotongan pada *video UCF Crime dataset* berdasarkan hasil klasifikasi objek penting.	Dihasilkan sistem yang dapat memperkecil besaran *video* dengan cara menghapus bagian *video* yang tidak penting dengan pendeteksian pergerakan manusia dan objek penting

3.2.4. Analisis Hasil Pengujian

Pada tahap ini penulis membahas tentang analisis hasil pengujian sistem pada tahap eksperimen dengan menggunakan *Confusion matrix* yang akan digunakan untuk menghitung *recall*, *precision* dan *F1-Score*. Perhitungan *recall, precision* dan *F1-Score* menggunakan persamaan (2.12), (2.13), (2.14) secara beruntun. Nilai evaluasi

berdasarkan prediksi sistem terhadap bagian-bagian pada *video* tentang terdapatnya gerakan objek penting atau tidak. Nilai *recall* yang tinggi dapat menunjukan nilai *False Positive* yang besar, dan untuk nilai *precision* yang tinggi menunjukan kehandalan model yang baik, lalu untuk nilai F1-Score akan memberikan gambaran seberapa seimbang nilai *recall* dan *precision*.

3.5. Metode Pengembangan/ Metode Pengukuran

Metode penelitian yang digunakan dalam penelitian ini adalah metode R&D (*Research and Development*). Metode penelitian dan pengembangan atau dalam bahasa Inggrisnya *Research and Development* adalah metode penelitian yang digunakan untuk menghasilkan produk tertentu, dan untuk menguji keefektifan produk tersebut penulis metode pengukuran *recall, precision* dan juga *F1-Score.*

3.6. Ilustrasi Perhitungan Metode

Pada bagian ini penulis akan melakukan simulasi perhitungan pada setiap metode yang akan penulis gunakan:

Tabel 3.2 Contoh *Confusion Matrix*

		Nilai Sebenarnya	
		Positif	Negatif
Nilai Prediksi Model	Positif	5	2
	Negatif	10	6

3.6.1. Perhitungan *Convolution*

Pada bagian ini penulis akan membahas tentang perhitungan operasi *convolution.* Didapatkan tensor input A dengan besar 4 x 4 dan filter W dengan besar 3 x 3 (*2.1*) serta nilai stride = 1 dan padding = 0:

$$A = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 3 & 1 & 3 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Maka hasil Y adalah :

$$Y_{[1][1]} = A_{[1,3]} \otimes W = \begin{bmatrix} 1 & 0 & 2 \\ 1 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$Y_{[1][1]} = 1 * 1 + 0 * 1 + 2 * 1 + 1 * 0 + 1 * 0 + 2 * 0 + 0 * 1 + 0 * 1 + 1 * 1 = 4$$

$$Y_{[1][2]} = A_{[1,3]} \otimes W = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$Y_{[1][1]} = 0 * 1 + 2 * 1 + 1 * 1 + 1 * 0 + 2 * 0 + 3 * 0 + 0 * 1 + 1 * 1 + 2 * 1 = 6$$

$$Y_{[2][1]} = A_{[1,3]} \otimes W = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 0 & 1 \\ 0 & 3 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$Y_{[1][1]} = 1 * 1 + 1 * 1 + 2 * 1 + 0 * 0 + 0 * 0 + 1 * 0 + 0 * 1 + 3 * 1 + 1 * 1 = 8$$

$$Y_{[2][2]} = A_{[1,3]} \otimes W = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 3 & 1 & 3 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$Y_{[1][1]} = 1 * 1 + 2 * 1 + 3 * 1 + 0 * 0 + 1 * 0 + 2 * 0 + 3 * 1 + 1 * 1 + 3 * 1 = 13$$

Maka didapat $Y = \begin{bmatrix} 4 & 6 \\ 8 & 13 \end{bmatrix}$

3.6.2. Perhitungan Pooling Layer

Pada bagian ini penulis akan membahas tentang perhitungan *pooling layer* yang dijelaskan pada 2.2.4.1. Didapatkan *tensor input* A dengan besar 3 x 3 maka operasi *pooling* dengan besar kernel = (2, 2), *stride* = (1, 1) dan juga metode *max* (*2.3*) adalah sebagai berikut:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$Y_{[1.1]} = Max(A_{[1,2],[1,2]}) = Max\left(\begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}\right) = 5$$

$$Y_{[1.2]} = Max(A_{[1,2],[2,3]}) = Max\left(\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}\right) = 6$$

$$Y_{[2.1]} = Max(A_{[2\ 3],[1,2]}) = Max\left(\begin{bmatrix}4 & 5\\ 7 & 8\end{bmatrix}\right) = 8$$

$$Y_{[2.2]} = Max(A_{[2,3],[2,3]}) = \ Max\left(\begin{bmatrix}5 & 6\\ 8 & 9\end{bmatrix}\right) = 9$$

Maka $Y = \begin{bmatrix}5 & 6\\ 8 & 9\end{bmatrix}$

3.6.3. Perhitungan Fungsi Aktifasi

Pada bagian ini penulis akan mensimulasikan perhitungan fungsi aktifasi pada CNN. Didapatkan sebuah *feature map* A sebesar 3 x 3.

$$A = \begin{bmatrix}-4 & -2 & -6\\ 2 & -5 & 10\\ 7 & 8 & -1\end{bmatrix}$$

Maka nilai A setelah dimasukan ke dalam fungsi aktifkasi ReLU adalah :

$$Y_{[1.1]} = Max(A_{[1.1]}, 0) = Max(-4, 0) = 0$$

$$Y_{[1.2]} = Max(A_{[1.2]}, 0) = Max(-2, 0) = 0$$

$$Y_{[1.3]} = Max(A_{[1.3]}, 0) = Max(-6, 0) = 0$$

$$Y_{[2.1]} = Max(A_{[2.1]}, 0) = Max(2.0) = 2$$

$$Y_{[2.2]} = Max(A_{[2.2]}, 0) = Max(-5, 0) = 0$$

$$Y_{[2.3]} = Max(A_{[2.3]}, 0) = Max(10, 0) = 10$$

$$Y_{[3.1]} = Max(A_{[3.1]}, 0) = Max(7, 0) = 7$$

$$Y_{[3.2]} = Max(A_{[3.2]}, 0) = Max(8, 0) = 8$$

$$Y_{[3.3]} = Max(A_{[3.3]}, 0) = Max(-1, 0) = 0$$

Maka $Y = \begin{bmatrix}0 & 0 & 0\\ 2 & 0 & 10\\ 7 & 8 & 0\end{bmatrix}$

3.6.4. Perhitungan *Correlation (CDDC)*

Pada tahapan ini penulis akan membahas tentang perhitungan yang terjadi pada tahap CDDC di algoritma fastflownet. Didapatkan sebuah CDDC layer dengan r = 0 serta $f_1 dan\ f_{warp}$ yang keduanya memiliki besaran 3 x 3.

$$f_1 = \begin{bmatrix}1 & 0 & 2\\ 1 & 1 & 2\\ 0 & 0 & 1\end{bmatrix}$$

$$f_{warp} = \begin{bmatrix}4 & 2 & 0\\ 1 & 0 & 2\\ 0 & 0 & 0\end{bmatrix}$$

$$c([1,1]) = 1 \times \frac{\begin{bmatrix}0 & 0 & 0\\ 0 & 4 & 2\\ 0 & 1 & 0\end{bmatrix}}{9} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0.44 & 0.22\\ 0 & 0.11 & 0\end{bmatrix}$$

$$c([1,2]) = 0 \times \frac{\begin{bmatrix}0 & 0 & 0\\ 4 & 2 & 0\\ 1 & 0 & 2\end{bmatrix}}{9} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$c([1,3]) = 2 \times \frac{\begin{bmatrix}0 & 0 & 0\\ 2 & 0 & 0\\ 0 & 2 & 0\end{bmatrix}}{9} = \begin{bmatrix}0 & 0 & 0\\ 0.44 & 0 & 0\\ 0 & 0.44 & 0\end{bmatrix}$$

$$c([2,1]) = 1 \times \frac{\begin{bmatrix}0 & 4 & 2\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}}{9} = \begin{bmatrix}0 & 0.44 & 0.22\\ 0 & 0.11 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$c([2,2]) = 1 \times \frac{\begin{bmatrix}4 & 2 & 0\\ 1 & 0 & 2\\ 0 & 0 & 0\end{bmatrix}}{9} = \begin{bmatrix}0.44 & 0.22 & 0\\ 0.11 & 0 & 0.22\\ 0 & 0 & 0\end{bmatrix}$$

$$c([2,3]) = 2 \times \frac{\begin{bmatrix}2 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix}}{9} = \begin{bmatrix}0.44 & 0 & 0\\ 0 & 0.44 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$c([3,1]) = 0 \times \frac{\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}{9} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

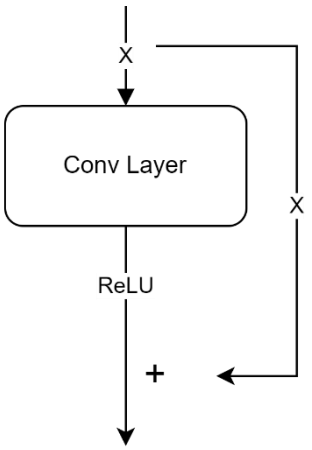
$$c([3,2]) = 0 \times \frac{\begin{bmatrix}1 & 0 & 2\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}{9} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$c([3,3]) = 1 \times \frac{\begin{bmatrix}0 & 2 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}{9} = \begin{bmatrix}0 & 0.22 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$c([1,1])$ sampai dengan $c([3,3])$ ditumpuk sehingga membentuk matriks *Cost Volume* dengan ukuran 3 x 3 x 9.

3.6.5. Perhitungan Pada *Residual Block*

Pada bagian ini, penulis akan membahas tentang perhitungan yang terjadi didalam *residual block.* Terdapat sebuah *residual block* dengan 1 layer (l_1) serta *identity map* X sebesar 3 x 3 yang melewati $l_1$dengan besar kernel W 2 x 2. Adapun arsitektur *residual block* yang dimaksut dapat diliat pada Gambar 3. 1.



Gambar 3. 1 Residual block untuk contoh perhitungan

$$X = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 3 & 1 & 3 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f(X,W)_{[1][1]} = X_{[1,3]} \otimes W = \begin{bmatrix} 1 & 0 & 2 \\ 1 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f(X,W)_{[1][1]} = 1 * 1 + 0 * 1 + 2 * 1 + 1 * 0 + 1 * 0 + 2 * 0 + 0 * 1 + 0 * 1 + 1 * 1 = 4$$

$$f(X,W)_{[1][2]} = X_{[1,3]} \otimes W = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f(X,W)_{[1][1]} = 0 * 1 + 2 * 1 + 1 * 1 + 1 * 0 + 2 * 0 + 3 * 0 + 0 * 1 + 1 * 1 + 2 * 1 = 6$$

$$f(X,W)_{[2][1]} = X_{[1,3]} \otimes W = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 0 & 1 \\ 0 & 3 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f(X,W)_{[1][1]} = 1 * 1 + 1 * 1 + 2 * 1 + 0 * 0 + 0 * 0 + 1 * 0 + 0 * 1 + 3 * 1 + 1 * 1 = 8$$

$$f(X,W)_{[2][2]} = X_{[1,3]} \otimes W = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 3 & 1 & 3 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f(X,W)_{[1][1]} = 1 * 1 + 2 * 1 + 3 * 1 + 0 * 0 + 1 * 0 + 2 * 0 + 3 * 1 + 1 * 1 + 3 * 1 = 13$$

$$f(X,W) = \begin{bmatrix} 4 & 6 \\ 8 & 13 \end{bmatrix}$$

Dikarenakan dimensi $f(X,W)$ tidak sama dengan nilai *identity map* (X) sehingga $f(X,W)$ perlu dilakukan *zero padding* untuk menyamai dimensi $f(X,W)$ dengan X.

$$f(X,W)' = \begin{bmatrix} 4 & 6 & 0 & 0 \\ 8 & 13 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

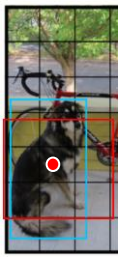
Setelah didapatkan $f(X,W)$ yang berdimensi sama dengan X, maka dapat melakukan operasi selanjutnya.

$$y = \begin{bmatrix} 4 & 6 & 0 & 0 \\ 8 & 13 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 3 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 6 & 2 & 1 \\ 9 & 14 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 3 & 1 & 3 \end{bmatrix}$$

Maka didapatkan y = $\begin{bmatrix} 5 & 6 & 2 & 1 \\ 9 & 14 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 3 & 1 & 3 \end{bmatrix}$

3.6.6. sssPerhitungan *Confidence Score Pada Bounding Box*

Pada bagian ini, penulis akan membahas tentang perhitungan *Confidence score untuk bounding box* pada algoritma YOLOv3. Pada Gambar 3.7 Terdapat sebuah *unit grid cell* yang dimana *vector* tengah dari objek pada citra berada pada *grid cell* tersebut yang berarti $P(Object) = 1$ [14]. Serta didapatkan nilai *Intersection* sebesar 700 dan nilai *Union* sebesar 1300. Maka perhitungan *Confidance Score* didapatkan sebagai berikut:



Gambar 3.7 Kasus perhitungan *confidence score* pada *bounding box*

$$Confidence = 1 * \frac{700}{1300} = 0.538$$

3.6.7. Perhitungan *Confidence Score Untuk Setiap Class*

Pada tahap ini penulis akan membahas tentang *Confidence Score* untuk setiap kelas yang akan diklasifikasikan. Didapatkan sebuah tensor output C yang berisi nilai dari 3 kelas yang akan diklasifikasikan oleh algoritma YOLOv3, yaitu C_1, C_2, C_3. Adapun nilai Intersection sebesar 1000 dan Union sebesar 1350. Maka proses perhitungan *Confidence score* (Z) sebagai berikut :

$$C = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.9 \end{bmatrix}$$

$$IOU = \frac{1000}{1350} = 0.74$$

Maka :

$$Y = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.9 \end{bmatrix} \times 0.74 = \begin{bmatrix} 0.222 \\ 0.296 \\ 0.666 \end{bmatrix}$$

3.6.8. Perhitungan Recall

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

Jika dimasukan nilai pada confusion matrix pada tabel 3.1 maka akan ditemukan nilai recall sebesar :

$$Recall = \frac{5}{5 + 6} = 0.45$$

3.6.9. Perhitungan Precision

$$Precision = \frac{TP}{TP + TN}$$

Jika dimasukan nilai pada confusion matrix pada tabel 3.1 maka akan ditemukan nilai precision sebesar :

$$Precision = \frac{5}{5 + 10} = 0.33$$

3.6.10. Perhitungan F1-Score

$$F1 = \frac{1}{\frac{1}{recall} + \frac{1}{precision}}$$

Jika dimasukan nilai precision dan juga recal pada bagian 3.6.1 dan 3.6.2 maka akan ditemukan nilai F-1 Score sebesar :

$$F1 = \frac{1}{\frac{1}{0.45} + \frac{1}{0.33}} = 0.190$$

DAFTAR PUSTAKA

[1] Universitas Prima Indonesia, R. Khairani, Universitas Prima Indonesia, and Y. Ariesa, 'PENGARUH KRIMINALITAS TERHADAP PERTUMBUHAN EKONOMI SUMATERA UTARA', *J. REP Ris. Ekon. Pembang.*, vol. 5, no. 2, pp. 166–178, Oct. 2020, doi: 10.31002/rep.v5i2.1954.

[2] B. U. Khan, M. B. Aziz, O. Faruk, and I. A. Talukder, 'Impact of CCTV Surveillance on Crime Prevention: A Study in Dhaka City', p. 13.

[3] A. M. Jansen, E. Giebels, T. J. L. van Rompay, and M. Junger, 'The Influence of the Presentation of Camera Surveillance on Cheating and Pro-Social Behavior', *Front. Psychol.*, vol. 9, p. 1937, Oct. 2018, doi: 10.3389/fpsyg.2018.01937.

[4] V. Prashyanusorn, P. Prashyanusorn, S. Kaviya, Y. Fujii, and P. P. Yupapin, 'The Use of Security Cameras with Privacy Protecting Ability', *Procedia Eng.*, vol. 8, pp. 301–307, 2011, doi: 10.1016/j.proeng.2011.03.056.

[5] 'Video Surveillance Storage: How Much Is Enough? | Seagate ASEAN', *Seagate.com*. https://www.seagate.com/as/en/solutions/surveillance/how-much-video-surveillance-storage-is-enough/ (accessed Aug. 04, 2022).

[6] F. Donald, C. Donald, and A. Thatcher, 'Work exposure and vigilance decrements in closed circuit television surveillance', *Appl. Ergon.*, vol. 47, pp. 220–228, Mar. 2015, doi: 10.1016/j.apergo.2014.10.001.

[7] S. Warsono Ibrahim, 'A comprehensive review on intelligent surveillance systems', *Commun. Sci. Technol.*, vol. 1, no. 1, 2016, doi: 10.21924/cst.1.1.2016.7.

[8] M. Zabłocki and K. Go, 'Intelligent video surveillance systems for public spaces – a survey', p. 16.

[9] M. Jouini, 'The impact of Artificial intelligence on surveillance camera system "Facial recognition growth"', *Am. J. Eng. Res.*, p. 6, 2020.

[10] R. Szeliski, 'Computer Vision: Algorithms and Applications, 2nd Edition', p. 1232.

[11] L. Alzubaidi *et al.*, 'Review of deep learning: concepts, CNN architectures, challenges, applications, future directions', *J. Big Data*, vol. 8, no. 1, p. 53, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[12] K. M. Lo, 'Optical Flow Based Motion Detection for Autonomous Driving'. arXiv, Mar. 02, 2022. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/2203.11693

[13] R. Ojala, J. Vepsäläinen, and K. Tammi, 'Motion detection and classification: ultra-fast road user detection', *J. Big Data*, vol. 9, no. 1, p. 28, Dec. 2022, doi: 10.1186/s40537-022-00581-8.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection'. arXiv, May 09, 2016. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/1506.02640

[15] L. Kong, C. Shen, and J. Yang, 'FastFlowNet: A Lightweight Network for Fast Optical Flow Estimation'. arXiv, Mar. 21, 2021. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/2103.04524

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, 'MobileNetV2: Inverted Residuals and Linear Bottlenecks'. arXiv, Mar. 21, 2019. Accessed: Nov. 02, 2022. [Online]. Available: http://arxiv.org/abs/1801.04381
[17] K. Sehairi, C. Fatima, and J. Meunier, 'Comparative study of motion detection methods for video surveillance systems', p. 69.
[18] J. Redmon and A. Farhadi, 'YOLOv3: An Incremental Improvement'. arXiv, Apr. 08, 2018. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/1804.02767
[19] M. G. Naftali, J. S. Sulistyawan, and K. Julian, 'Comparison of Object Detection Algorithms for Street-level Objects'. arXiv, Aug. 24, 2022. Accessed: Nov. 02, 2022. [Online]. Available: http://arxiv.org/abs/2208.11315
[20] W. Sultani, C. Chen, and M. Shah, 'Real-world Anomaly Detection in Surveillance Videos'. arXiv, Feb. 14, 2019. Accessed: Nov. 02, 2022. [Online]. Available: http://arxiv.org/abs/1801.04264
[21] Direktorat Statistik Ketahanan Sosial, *STATISTIK KRIMINAL 2021*, vol. 2021. Badan Pusat Statistik.
[22] F. R. Doni and A. M. Lukman, 'Rancang Bangun Sistem Monitoring Kamera CCTV Online Dengan Penerapan Hik-Connet', *EVOLUSI: Jurnal Sains dan Manajemen*, vol. 9, no. 1, Mar. 2021, [Online]. Available: https://ejournal.bsi.ac.id/ejurnal/index.php/evolusi/article/download/9984/4768
[23] Mrs. N. Farhat and M. Nasiruddin, 'Review on Camera based Surveillance Systems', *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 3, pp. 1477–1479, Mar. 2022, doi: 10.22214/ijraset.2022.40882.
[24] I. Kokadwar, A. Kulkarni, S. Khare, V. Limbhore, and S. Chandurkar, 'Camera based Smart Surveillance System-Literature Survey', vol. 07, no. 06, p. 3, 2020.
[25] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[26] K. Das, R. N. Behera, and B. Tech, 'A Survey on Machine Learning: Concept, Algorithms and Applications', vol. 5, no. 2, p. 10, 2007.
[27] Dr. A. Bashar, 'SURVEY ON EVOLVING DEEP LEARNING NEURAL NETWORK ARCHITECTURES', *J. Artif. Intell. Capsule Netw.*, vol. 2019, no. 2, pp. 73–82, Dec. 2019, doi: 10.36548/jaicn.2019.2.003.
[28] A. Amini, 'Deep Computer Vision'. Accessed: Nov. 02, 2022. [Online]. Available: http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L3.pdf
[29] J. Chai, H. Zeng, A. Li, and E. W. T. Ngai, 'Deep learning in computer vision: A critical review of emerging techniques and application scenarios', *Mach. Learn. Appl.*, vol. 6, p. 100134, Dec. 2021, doi: 10.1016/j.mlwa.2021.100134.
[30] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, 'Convolutional neural networks: an overview and application in radiology', *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.
[31] K. O'Shea and R. Nash, 'An Introduction to Convolutional Neural Networks', p. 12.
[32] S. Singh, U. Ahuja, M. Kumar, K. Kumar, and M. Sachdeva, 'Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment', *Multimed. Tools Appl.*, vol. 80, no. 13, pp. 19753–19768, May 2021, doi: 10.1007/s11042-021-10711-8.
[33] S. Neelam, 'YOLO for Object Detection, Architecture Explained!', *Analytics Vidhya*, Jan. 19, 2022. https://medium.com/analytics-vidhya/understanding-

yolo-and-implementing-yolov3-for-object-detection-5f1f748cc63a (accessed Oct. 22, 2022).
[34] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, 'Feature Pyramid Networks for Object Detection'. arXiv, Apr. 19, 2017. Accessed: Nov. 02, 2022. [Online]. Available: http://arxiv.org/abs/1612.03144
[35] K. He, X. Zhang, S. Ren, and J. Sun, 'Deep Residual Learning for Image Recognition'. arXiv, Dec. 10, 2015. Accessed: Nov. 02, 2022. [Online]. Available: http://arxiv.org/abs/1512.03385
[36] X. Zhang, Y. Gao, H. Wang, and Q. Wang, 'Improve YOLOv3 using dilated spatial pyramid module for multi-scale object detection', *Int. J. Adv. Robot. Syst.*, vol. 17, no. 4, p. 172988142093606, Jul. 2020, doi: 10.1177/1729881420936062.
[37] P. Fischer *et al.*, 'FlowNet: Learning Optical Flow with Convolutional Networks'. arXiv, May 04, 2015. Accessed: Nov. 02, 2022. [Online]. Available: http://arxiv.org/abs/1504.06852
[38] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, 'PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume'. arXiv, Jun. 25, 2018. Accessed: Nov. 02, 2022. [Online]. Available: http://arxiv.org/abs/1709.02371
[39] X. Zhang, X. Zhou, M. Lin, and J. Sun, 'ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices'. arXiv, Dec. 07, 2017. Accessed: Nov. 28, 2022. [Online]. Available: http://arxiv.org/abs/1707.01083
[40] A. Paszke *et al.*, 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. arXiv, Dec. 03, 2019. Accessed: Sep. 11, 2022. [Online]. Available: http://arxiv.org/abs/1912.01703
[41] K. R. Srinath, 'Python – The Fastest Growing Programming Language', vol. 04, no. 12, p. 4.
[42] T. C. A.-S. Zulkhaidi, E. Maria, and Y. Yulianto, 'Pengenalan Pola Bentuk Wajah dengan OpenCV', *J. Rekayasa Teknol. Inf. JURTI*, vol. 3, no. 2, p. 181, Jun. 2020, doi: 10.30872/jurti.v3i2.4033.
[43] Z. Karimi, 'Confusion Matrix', p. 5, Oct. 2021.
[44] F. Gozali and N. Akbar, 'PENDETEKSIAN DAN PEREKAMAN GERAKAN BENDA DALAM SISTEM PENGAWASAN KEAMANAN DENGAN MENGGUNAKAN RASPBERRY PI', *Jetri J. Ilm. Tek. Elektro*, Feb. 2017, doi: 10.25105/jetri.v11i2.1442.
[45] A. N. A. Thohari and R. D. Ramadhani, 'Sistem Pengawasan Berbasis Deteksi Gerak Menggunakan Single Board Computer', *J. Nas. Tek. Elektro Dan Teknol. Inf. JNTETI*, vol. 8, no. 1, p. 1, Mar. 2019, doi: 10.22146/jnteti.v8i1.483.