

Pembelajaran Kolaboratif Berdasarkan *Two-Branch Neural Network* dan YOLOv5 Untuk Deteksi Objek Pada Kendaraan Otonom

1st Agniya Tazkiya Aulia
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia

agniyat@student.telkomuniversity.ac.id

2nd Suryo Adhi Wibowo
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia

suryoadhiwibowo@telkomuniversity.ac.id

3rd Fityanul Akhyar
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia

fityanul@telkomuniversity.ac.id

Abstrak—Seiring dengan kemajuan teknologi dan otomatisasi, perkembangan pada *Autonomous Vehicle* (AV) meningkat secara signifikan. *Object detection* memegang peranan penting pada teknologi AV. Pada penerapannya, kondisi cuaca yang buruk mengakibatkan terjadinya penurunan performa sistem dalam mendeteksi objek terutama ketika cuaca berkabut. Tugas Akhir ini menganalisis konfigurasi dari pembelajaran kolaboratif antara algoritma *dehazing* dan *object detection* untuk meningkatkan kinerja sistem AV dalam mendeteksi objek di kondisi cuaca berkabut. Algoritma *dehazing* yang digunakan adalah *Two-Branch Neural Network*, sedangkan algoritma *object detection* yang digunakan adalah YOLOv5. Pada YOLOv5 dilakukan optimasi dengan *hyperparameter tuning* untuk mendapatkan nilai pengukuran terbaik. Hasil penelitian menunjukkan bahwa model pembelajaran kolaboratif memiliki mAP yang lebih tinggi dari model YOLOv5 orisinal, dengan nilai 71,5%. Di sisi lain, konfigurasi *hyperparameter* terbaik didapatkan pada nilai *learning rate* 0,00334; *batch size* 32; dan lainnya didapatkan dari *hyperparameter* VOC. Hal ini meningkatkan mAP dari 71,5% ke 74,8%.

Kata kunci—AV, YOLOv5, *two-branch neural network*, *object detection*, *image dehazing*, *hyperparameter*

I. PENDAHULUAN

Seiring dengan kemajuan teknologi dan otomatisasi, perkembangan *Smart Surveillance System* (SSS) mengalami peningkatan yang signifikan khususnya pada *Autonomous Vehicle* (AV). AV dirancang dengan penggabungan dari berbagai teknologi, salah satunya yang memegang peran penting yaitu *object detection* [1]. *Convolutional Neural Network* (CNN) umum digunakan sebagai algoritma utama pada sistem AV yang berfungsi untuk mengenali dan mengklasifikasi objek di lingkungan sekitar. Namun, pada penerapannya, kondisi cuaca yang buruk mengakibatkan terjadinya penurunan performa sistem untuk mendeteksi objek terutama ketika cuaca berkabut [2]. Maka dari itu, dibutuhkan pendekatan solutif guna meningkatkan performa sistem AV dalam mendeteksi objek pada kondisi cuaca berkabut.

Image dehazing merupakan salah satu solusi efektif ketika menghadapi masalah restorasi pada gambar berkabut. Pada penelitian Li dkk., disebutkan bahwa ketika menggabungkan model Faster-RCNN dengan AOD-Net, terdapat peningkatan

parameter *mean Average Precision* (mAP) sebesar 4,54% pada kondisi kabut tipis, 5,88% pada kondisi kabut sedang, dan 12,39% pada kondisi kabut tebal [3]. Hal ini menunjukkan adanya peningkatan performa deteksi objek dengan menggabungkan model CNN dengan *image dehazing*. Tetapi, beberapa *image dehazing* memiliki keterbatasan dalam menghilangkan kabut yang tidak terdistribusi secara merata atau disebut dengan kabut non-homogen. Sedangkan, kabut yang sering dijumpai di kehidupan nyata berupa kabut non-homogen. Yu dkk. memperkenalkan *Two-Branch Neural Network* yang mampu menghilangkan kabut non-homogen menggunakan metode *ensemble learning* dan berhasil mengungguli model *image dehazing* lainnya yang diuji dengan dataset NTIRE2021 menghasilkan PSNR dan SSIM sebesar 37,61 dan 0,991 [4]. Jika dibandingkan berdasarkan kecepatannya, *Two-Branch Neural Network* berada di urutan ke-2 setelah AOD-Net, meskipun dengan jumlah parameter yang terbesar [4].

Bagaimanapun, dengan hanya menggabungkan model deteksi objek dan *image dehazing* saja, tidak akan selalu menjamin peningkatan performa dan terlebih lagi akan memperlambat kecepatan deteksi [5]. Untuk mengatasi permasalahan tersebut, Huang dkk. mengusulkan metode Dual-Subnet Network (DSNet) berbasis *multi-task learning* yang terdiri dari *feature restoration* (FR) *subnetwork* dan RetinaNet sebagai *object detection subnetwork* [5]. DSNet dilatih secara *end-to-end* menggunakan FOD Dataset agar dapat mempelajari peningkatan visibilitas, *object detection*, dan *object localization* secara bersamaan [5]. Pada penelitian ini, diketahui bahwa dengan menggunakan *restoration subnetwork* pada model DSNet, terjadi peningkatan mAP yaitu menjadi 58,01% pada objek *car* dan 43,67% pada objek *person* [5]. Meskipun penelitian tersebut telah mendapatkan hasil yang terbaik dibandingkan dengan metode lainnya, nilai mAP dari DSNet masih rendah. Penyebab dari masalah tersebut adalah rendahnya mAP dari *backbone* RetinaNet. Peningkatan mAP dapat dicapai dengan mengimplementasikan You Only Look Once (YOLO)v5 [6], algoritma deteksi objek terbaru yang memiliki tingkat akurasi tinggi.

Berdasarkan data tersebut, penulis mengusulkan penelitian

lebih lanjut untuk meningkatkan performa deteksi objek pada kondisi cuaca berkabut dengan menggabungkan algoritma *dehazing* dan *object detection* sebagai Tugas Akhir. Penelitian ini dilakukan dengan menggunakan *Two-Branch Neural Network* untuk proses *dehazing* dan YOLOv5 untuk proses deteksi objek. Penulis juga melakukan *hyperparameter tuning* pada model YOLOv5 untuk meningkatkan nilai mAP dan mengoptimalkan kinerja deteksi. Model akan dilatih untuk dapat mendeteksi objek *car* dan *person* menggunakan *dataset* Foggy Cityscapes dan diuji dengan *dataset* Foggy Driving [7]. Setelah itu, hasil dari penelitian ini akan melalui proses analisis, evaluasi, dan perbandingan parameter performansi dengan beberapa model lainnya.

II. KAJIAN TEORI

A. You Only Look Once (YOLO)

You Only Look Once (YOLO) adalah algoritma *object detection* yang arsitekturnya didesain untuk dapat memproses gambar dengan kecepatan *real-time* yaitu 45 *frame per second* [8]. Jaringan deteksi YOLO dibangun dengan 24 *convolutional layer* dan dua *fully-connected layer*. Selain itu, YOLO menggunakan kerangka kerja *Darknet* pada setiap proses *training* dan *inference*. Salah satu fitur penting dari jaringan berbasis YOLO adalah penggunaan *grid* dengan ukuran tertentu yang membagi input gambar menjadi beberapa *cell*. Masing-masing *grid cell* bertanggung jawab untuk tugas deteksi objek, yaitu keluaran berupa probabilitas kelas dan *bounding box* beserta *confidence*-nya [9].

1. *Fungsi Loss*: YOLO menggunakan SSE (*Sum-Squared Error*) sebagai *backbone* pada fungsi *loss*. Pada beberapa gambar, banyak *grid cell* yang tidak terdapat objek didalamnya dengan nilai *confidence score* nol. Hal ini dapat melemahkan perhitungan *loss* pada *cell* yang memiliki objek didalamnya. YOLO menghindari masalah ketidakstabilan tersebut dengan meningkatkan nilai *loss* pada koordinat *bounding box* prediksi saat terdapat objek ($\lambda_{coord} = 5$), dan mengurangi nilai *loss* pada prediksi saat tidak ada objek ($\lambda_{noobj} = 0.5$). Fungsi *loss* YOLO dihitung melalui penjumlahan parameter-parameter *loss* pada *bounding box*, yaitu koordinat (x,y,w,h), *confidence score*, dan probabilitas kelas, yang ditunjukkan pada Persamaan 1 [10].

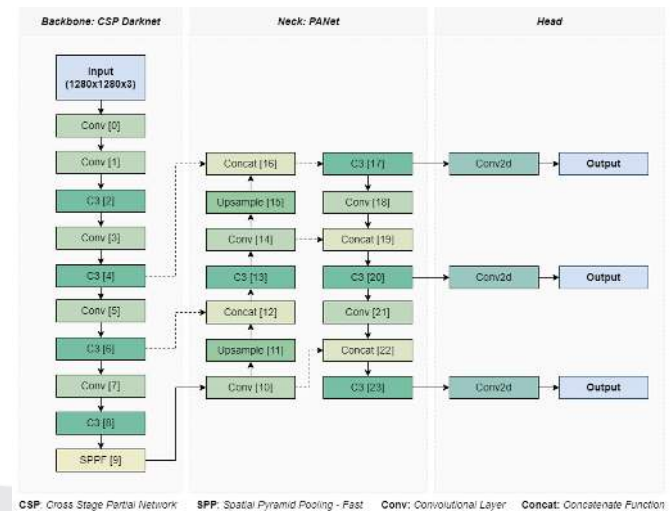
$$\begin{aligned} \mathcal{L} = & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \quad (1)$$

Nilai $\mathbb{1}_{ij}^{obj}$ akan sama dengan 1 ketika terdapat objek, dan akan sama dengan 0 ketika tidak terdapat objek. Sedangkan nilai dari $\mathbb{1}_{ij}^{noobj}$ berlaku sebaliknya. Bagian pertama dan

kedua dari persamaan tersebut mendefinisikan *loss* dari koordinat (x,y) serta tinggi dan lebar (w,h) *bounding box* prediksi. Pada bagian ketiga, dilakukan perhitungan terhadap *confidence score* saat terdapat objek maupun tidak. Bagian terakhir dari persamaan menghitung *loss* terhadap *error* pada probabilitas kelas.

B. YOLOv5

YOLOv5 termasuk dalam keluarga dari YOLO (You Only Look Once) dan dibagi menjadi empat model yang dengan ukuran arsitektur berbeda, diantaranya adalah YOLOv5s, YOLOv5m, YOLOv5l, dan YOLOv5x. Jaringan struktur YOLOv5 masih dekat dengan YOLOv4 [11], ilustrasinya dapat dilihat pada Gambar 1. YOLOv5 menggunakan *Cross Stage Partial Network* (CSPNet) [12] sebagai *backbone* untuk menghasilkan kombinasi gradien sekaligus mengurangi proses komputasi. Pada bagian *neck*, arsitektur ini menggunakan *Spatial Pyramid Pooling-Fast* (SPPF) *block* dan struktur dari PANet (*Path Aggregation Network*) dengan fitur piramida berdasarkan FPN (*Feature Pyramid Network*) *Layer*. Bagian *head* YOLOv5 mengeluarkan prediksi berupa vektor yang berisi prediksi kelas dan koordinat *bounding box* beserta *confidence score*-nya.



GAMBAR 1
(Arsitektur YOLOv5)

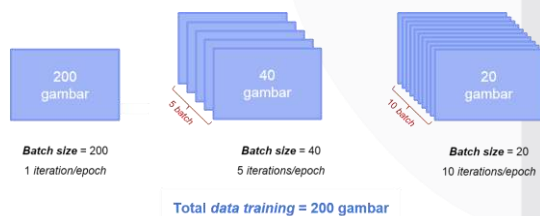
C. Hyperparameter Tuning

Hyperparameter tuning adalah proses untuk merancang model *machine learning* yang ideal dengan konfigurasi *hyperparameter* optimal [13]. *Hyperparameter* berfungsi untuk melakukan konfigurasi pada model *machine learning* dan menentukan algoritma yang digunakan untuk memperkecil fungsi *loss* [14]. *Hyperparameter tuning* dianggap sebagai komponen penting untuk membangun model *machine learning* yang efektif. Tipe *hyperparameter* pada setiap model berbeda-beda. Pada model YOLOv5, parameter yang dapat diubah guna meningkatkan performa model adalah ukuran gambar input, epochs, *batch size*, *learning rate*, momentum, dan fungsi aktivasi. Parameter-parameter ini ditetapkan sebelum proses

training dimulai. Salah satu cara yang digunakan untuk *tune hyperparameter* adalah *manual testing*. Cara ini dilakukan dengan menguji dan mengevaluasi model pada setiap nilai parameter yang telah ditentukan.

1) *Learning Rate*: *Hyperparameter* latih pada model *machine learning* yang bertujuan untuk menghitung nilai koreksi bobot pada proses pelatihan disebut dengan *Learning Rate* (LR). Dengan kata lain, LR menentukan besarnya *step* pada setiap iterasi untuk mencapai lokal minimum. Penentuan nilai LR memiliki pengaruh yang besar terhadap akurasi dan kecepatan latih pada model [15]. Nilai LR yang terlalu besar akan menyebabkan perubahan variabel yang terlalu jauh, sehingga menyebabkan koreksi nilai yang berlebihan. Hal ini berpengaruh pada berkurangnya akurasi dan ketelitian jaringan dalam mencapai solusi optimal yang diinginkan. Namun, nilai LR yang tinggi memungkinkan model untuk mencapai konvergensi dan mempercepat proses pembelajaran. Mengurangi nilai LR dapat menghindari koreksi bobot yang berlebihan sehingga proses pembelajaran menjadi lebih stabil. Ketika nilai LR cukup kecil, akurasi pada model akan meningkat dengan konsekuensi *steps* yang dibutuhkan semakin banyak dan memperlambat proses pembelajaran [16].

2) *Batch Size*: *Batch size* menentukan jumlah sampel yang akan dilatih oleh jaringan sebelum parameter model diperbarui. Pada akhir setiap *batch*, nilai *loss* akan dihitung dan parameter model akan diperbarui berdasarkan nilai tersebut dengan tujuan meningkatkan performa model. *Batch size* mempengaruhi beberapa indikator seperti akurasi dan waktu komputasi [17]. Ketika sebuah jaringan tidak membagi *dataset* menjadi beberapa *batch*, *dataset* akan dipelajari secara keseluruhan sebelum bobot dikoreksi. Hal ini akan meminimalisasi fungsi *loss* karena jaringan telah melihat seluruh sampel *dataset* sebelum membuat perubahan pada model. Namun, memori yang dibutuhkan akan lebih besar dibanding membagi *dataset* ke dalam beberapa *batch*. Gambar 2 menunjukkan contoh penyebaran data *training* yang dibagi ke dalam beberapa *batch*.

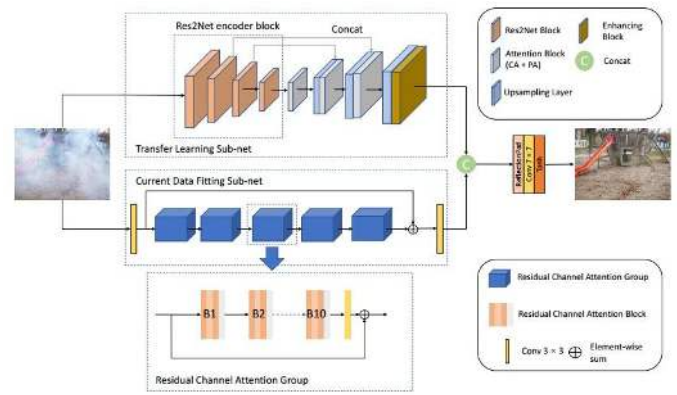


GAMBAR 2

(Contoh pembagian data sesuai ukuran *batch*)

D. Two-Branch Neural Network

Two-Branch Neural Network adalah metode penghilang kabut non-homogen natural pada citra dengan *ensemble learning* [4]. Metode ini terdiri dari dua *sub-network*, yaitu *transfer learning sub-net* dan *current data fitting sub-net*. Masing-masing *sub-net* memiliki tujuan yang berbeda, yaitu *transfer learning* digunakan untuk mengekstraksi representasi global



GAMBAR 3

(Arsitektur *Two-Branch Neural Network* [4])

dari gambar hasil *pre-trained* dan *current data fitting sub-net* digunakan untuk bekerja pada data saat ini agar mendapatkan performa yang baik pada gambar *training* yang spesifik. *Transfer learning sub-net* menggunakan Res2Net [18] sebagai *encoder* sedangkan *current data fitting sub-net* dilatih dengan *dataset* NTIRE 2021 NH-Haze. *Fusion tail* pada metode ini digunakan untuk menggabungkan fitur dari kedua cabang, hal ini lah yang disebut dengan *ensemble learning*. Arsitektur dari *Two-Branch Neural Network* dapat dilihat pada Gambar 3.

E. Evaluasi Model

Salah satu metrik yang umum dijadikan tolak ukur untuk menganalisis suatu model *object detection* yaitu *Mean Average Precision* (mAP). Penggunaan mAP sebagai parameter performansi bertujuan mengevaluasi *robustness* dari sebuah model. Selain itu, mAP menggabungkan *trade-off* antara nilai *precision* dan *recall* sehingga dapat memaksimalkan efek dari kedua metrik tersebut.

1) *Mean Average Precision*: *Precision* adalah kemampuan model dalam mendeteksi objek yang relevan atau presentase dari prediksi positif yang benar. Sedangkan *Average Precision* (AP) diambil dari nilai *precision* dan *recall*. Perhitungan AP ditujukan untuk masing-masing kelas. Pada *dataset* yang besar, dibutuhkan metrik yang mampu mempresentasikan keakuratan model dalam mendeteksi semua kelas. Dalam kasus ini, mAP yang merupakan rata-rata AP dari semua kelas, menjadi metrik yang dibutuhkan. Perhitungan mAP dan AP didefinisikan pada Persamaan 2 dan 3.

$$mAP = \frac{1}{C} \sum_{i=1}^C AP_i \quad (2)$$

$$AP = \sum_n [R_n - R_{n-1}] P_n \quad (3)$$

Pada mAP, *C* merupakan jumlah kelas dan AP merupakan *Average Precision* dari tiap kelas. Pada AP, *R_n* dan *P_n* merupakan *recall* dan *precision* pada saat *threshold* ke-*n*. Sedangkan, nilai *recall* dan *precision* dihitung berdasarkan nilai *true positive* (TP), *false positive* (FP), dan *false negative*

(FN). Perhitungan dari *recall* dan *precision* ditunjukkan pada Persamaan 4 dan 5.

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (5)$$

III. METODE

A. Diagram Alir Penelitian



GAMBAR 4
(Diagram alir penelitian)

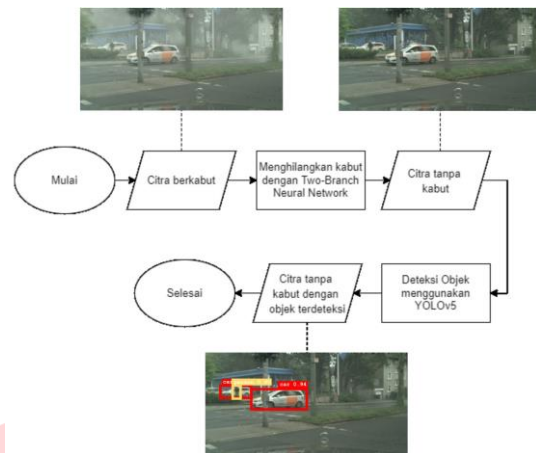
Diagram alir penelitian Tugas Akhir ini menjadi panduan bagi penulis dalam merancang sistem pembelajaran kolaboratif untuk kendaraan otonom pada kondisi lingkungan berkabut. Peningkatan kualitas citra berkabut pada *dataset* Foggy Cityscapes dilakukan dengan proses *image dehazing* menggunakan model *Two-Branch Neural Network*. Kemudian, *dataset* tersebut menjadi masukan untuk melatih YOLOv5 yang dioptimalkan dengan proses *hyperparameter tuning*. Diagram alir dapat dilihat pada Gambar 4.

B. Konfigurasi Model Collaborative Learning

Pada penelitian ini, terdapat skema konfigurasi model yang menggabungkan antara YOLOv5 sebagai *object detection* dan *Two-Branch* sebagai *image dehazing*. Penggabungan antara algoritma kedua model ini disebut dengan *collaborative learning*, bertujuan untuk mendapatkan kinerja terbaik dari *object detection* pada sistem *autonomous vehicle*. Diagram alir dari skema tersebut dapat dilihat pada Gambar 5. Keluaran yang dihasilkan berupa citra tanpa kabut dengan *bounding box* dan kelas dari objek yang diprediksi.

C. Dataset

Dataset yang digunakan pada penelitian ini adalah *dataset* Foggy Cityscapes [19]. *Dataset* ini terdiri dari 5.000 gambar pemandangan 21 kota di Jerman yang diambil dari sudut pandang pengemudi kendaraan. *Dataset* tersebut kemudian



GAMBAR 5
(Skema konfigurasi model Collaborative Learning)



(a) Cerah (b) 600 m (c) 300 m (d) 150 m

GAMBAR 6
(Dataset Foggy Cityscapes [7])

dikembangkan menjadi 15.000 gambar dengan kabut sintetik yang memiliki tiga level jarak pandang, yaitu sejauh 600 m, 300 m, dan 150 m. Terdapat delapan kelas objek yang telah dilabeli dengan anotasi *bounding box* pada *dataset* ini, diantaranya adalah *person*, *car*, *truck*, dan lain sebagainya. Namun, penelitian ini hanya menggunakan dua kelas, yaitu *car* dan *person*. Beberapa sampel *dataset* Foggy Cityscapes dapat dilihat pada Gambar 6. *Dataset* dipecah ke dalam tiga bagian, yaitu *training*, *validation*, dan *test*. Bagian *dataset test* pada penelitian ini tidak digunakan dan diganti dengan *dataset* Foggy Driving. Statistik *dataset* Foggy Cityscapes tanpa bagian *test* ditunjukkan pada Tabel I.

TABEL I
(Statistik dataset Foggy Cityscapes)

Set	Jumlah Gambar	Jumlah Objek	
		Car	Person
Training	8.925	83.712	59.196
Validation	1.500	14.571	11.862
Total	10.425	98.283	71.058

Selain melatih model *object detection*, *dataset* juga dibutuhkan untuk menguji model. Dalam tahap pengujian, digunakan *dataset* Foggy Driving. Berbeda dengan Foggy Cityscapes, *dataset* Foggy Driving merupakan *dataset* yang terdiri dari 101 gambar pemandangan kota dengan kondisi berkabut. *Dataset* ini dikumpulkan dan dianotasi dengan tujuan murni untuk menguji kemampuan *object detector* yang

dilatih dengan gambar berkabut sintetis dalam memahami gambar dengan kabut asli di kehidupan nyata. Beberapa sampel *ground-truth dataset* Foggy Driving dapat dilihat pada Gambar 7.



GAMBAR 7
(Dataset Foggy Driving [7])

D. Image Dehazing Dengan Two-Branch Neural Network

Sebelum dilakukan proses *training* dan *testing* pada model objek deteksi, yaitu YOLOv5, dilakukan proses *image dehazing* atau penghilangan kabut pada gambar *dataset* Foggy Cityscapes dan Foggy Driving. Proses ini dilakukan sebagai langkah *pre-processing* untuk meningkatkan kualitas gambar agar visibilitas pada gambar menjadi lebih baik. Penghilangan kabut pada gambar dilakukan dengan menggunakan model *pre-trained* dari Two-Branch Neural Network. Metode ini telah terbukti efektif dalam menghilangkan kabut non-homogen atau kabut alami yang ketebalannya tidak merata [4].

E. Training YOLOv5

Proses *training* dilakukan dengan metode *transfer learning*, yaitu melatih model yang sudah dilatih terlebih dahulu menggunakan kumpulan dataset atau disebut dengan model *pre-trained*. Metode ini bertujuan untuk memanfaatkan pengetahuan yang telah didapat oleh model dari pembelajaran sebelumnya, sehingga tidak perlu melatih model dari awal. Model *pre-trained* YOLOv5 dilatih dengan *dataset* COCO yang didalamnya terdapat 80 kelas objek. *Pre-trained* model ini dijadikan sebagai titik awal untuk *training* dengan *dataset* baru. Setiap model memiliki keunggulannya masing-masing baik dari sisi kecepatan maupun akurasi. Model yang sesuai dengan kebutuhan penelitian adalah YOLOv5s yang unggul dalam kecepatan dengan mempertahankan akurasi dibandingkan model lainnya.

Setelah menentukan model, model tersebut dilatih dengan masukan *dataset* Foggy Cityscapes. Kemudian dilakukan penyesuaian *hyperparameter*, yaitu *input size*, *learning rate*, *batch size*, jumlah *epoch*, dan *hyperparameter* lainnya. Angka dari masing-masing *hyperparameter* akan memengaruhi kinerja model *object detection*. *Hyperparameter* yang digunakan dalam melatih model *pre-trained* YOLOv5 dapat dilihat pada Tabel II. Terdapat dua referensi *hyperparameter* yang diuji pada penelitian ini, yaitu *hyperparameter* bawaan YOLOv5 dan *hyperparameter* VOC.

F. Spesifikasi Perangkat

Perangkat keras yang digunakan pada penelitian ini merupakan laptop yang didukung server NVidia DGX A100. Penggunaan server yang dihubungkan dengan laptop melalui *command prompt* memiliki tujuan untuk meningkatkan kecepatan komputasi dalam melatih dan menguji model.

TABEL II
(Masukan *hyperparameter training* YOLOv5)

Hyperparameter	Nilai
Input Size	640 × 1280 px
Learning Rate	0,1; 0,01; 0,00334*
Batch Size	16, 32
Epoch	100
Others	Bawaan YOLOv5, VOC

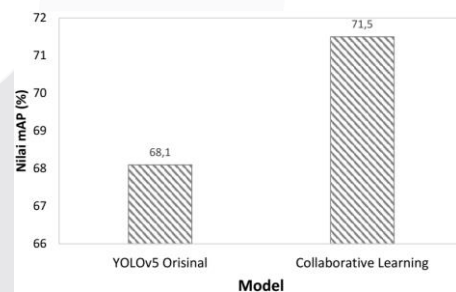
*Learning rate bawaan Hyperparameter VOC

Selain itu, proses latih, validasi, dan uji model dilakukan menggunakan platform JupyterLab. Platform ini menyediakan beberapa fitur untuk pengguna agar dapat menjalankan aktivitas pengkodean, seperti Jupyter Notebook, teks editor, terminal, dan komponen lain yang dapat disesuaikan dengan fleksibel. Pada penelitian ini, bahasa pemrograman yang digunakan adalah Python. Python memiliki berbagai macam *framework* dan *library* yang mendukung pada *train* model dan analisa data.

IV. HASIL DAN PEMBAHASAN

A. Analisis Perbandingan Antara Model YOLOv5 Orisinal dan Collaborative Learning

Bagian ini menunjukkan pengaruh penggunaan Two-Branch Neural Network sebagai *image dehazing subnet* dengan tujuan meningkatkan performa objek detektor, yaitu YOLOv5, pada kondisi lingkungan berkabut. Validasi hasil latih model yang dilakukan pada *validation dataset* menghasilkan mAP seperti yang ditunjukkan pada Gambar 8. Berdasarkan grafik tersebut, model Collaborative Learning (CL) memiliki mAP yang lebih tinggi dibandingkan dengan model YOLOv5 orisinal dengan selisih 3,4%. Hal ini membuktikan adanya peningkatan performa YOLOv5 ketika kabut pada gambar dihilangkan menggunakan Two-Branch Neural Network.



GAMBAR 8
(Grafik perbandingan nilai mAP antara model YOLOv5 orisinal dan Collaborative Learning)

Percobaan deteksi objek pada *test dataset* Foggy Driving menggunakan model CL menunjukkan skor konfidensi yang lebih tinggi. Hal ini dapat dilihat pada Gambar 9. Pada gambar di baris kedua, model YOLOv5 orisinal gagal untuk mendeteksi masing-masing satu objek *car* dan *person* yang berhasil dideteksi oleh model CL. Kegagalan deteksi objek *car* pada model YOLOv5 orisinal juga dapat dilihat pada gambar di baris ketiga. Ketiga percobaan menunjukkan bahwa model CL mengungguli YOLOv5 orisinal dari segi skor konfidensi.



GAMBAR 9

(Hasil deteksi pada model: (a) YOLOv5 orisinal dan (b) *Collaborative Learning*)

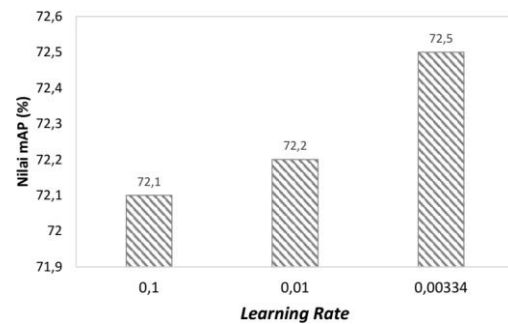
Hal tersebut membuktikan bahwa model CL dapat lebih baik dalam mendeteksi objek pada citra dengan kabut alami.

Hasil percobaan ini menunjukkan bahwa keberadaan kabut berdampak pada penurunan performa sistem ditandai dengan adanya kesalahan deteksi serta skor konfidensi dan nilai mAP yang rendah. Hal tersebut disebabkan oleh penglihatan objek detektor yang buruk karena terhalang kabut, sehingga ekstraksi fitur pada citra menjadi tidak maksimal. Maka dari itu, penambahan algoritma *image dehazing* untuk memulihkan citra berkabut sebagai masukan pada objek detektor menjadi sangat penting. Hal ini dibuktikan oleh hasil perbandingan evaluasi model YOLOv5 orisinal dan CL secara kuantitatif dan kualitatif yang menunjukkan peningkatan kinerja model dalam mendeteksi objek pada citra berkabut.

B. Konfigurasi Learning Rate pada Model YOLOv5

Pada proses ini, terdapat tiga masukan *learning rate* yang diuji untuk melatih model YOLOv5, yaitu 0,1; 0,01; dan 0,00334. Sedangkan, parameter latih selain *learning rate* diatur dengan angka yang sama. *Mean average precision* tertinggi dengan nilai 72,5% dicapai oleh *learning rate* terkecil yaitu 0,00334. Perbandingan mAP terhadap ketiga *learning rate* dapat dilihat pada Gambar 10.

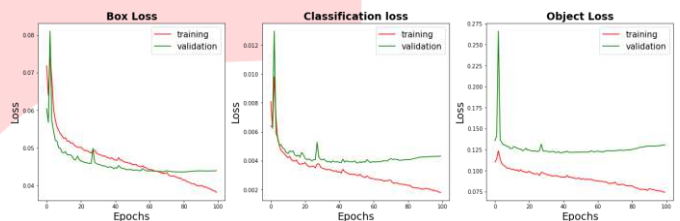
Pada grafik tersebut, ditunjukkan bahwa model dengan *learning rate* 0,00334 mengungguli model lain dengan selisih mAP sebesar 0,4% dan 0,3%. Dari data tersebut, dapat diketahui bahwa besar nilai LR berbanding lurus dengan akurasi pada model. Hal ini membuktikan bahwa mengurangi nilai LR merupakan salah satu cara untuk meningkatkan akurasi. Penyebabnya adalah nilai LR yang tinggi mengakibatkan koreksi bobot pada model terlalu berlebihan, sehingga ketelitian



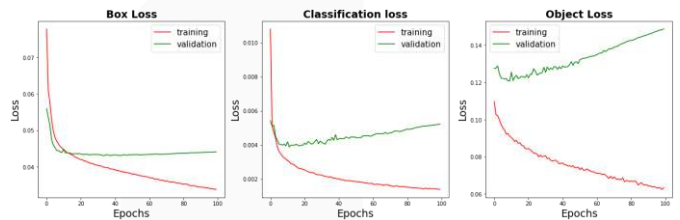
GAMBAR 10

Grafik perbandingan nilai mAP pada konfigurasi *learning rate*

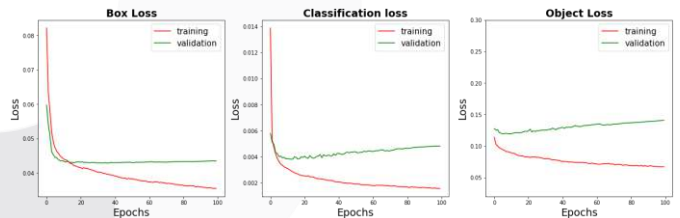
model dalam mencapai solusi optimal yang diinginkan akan berkurang.



(a)



(b)



(c)

GAMBAR 11

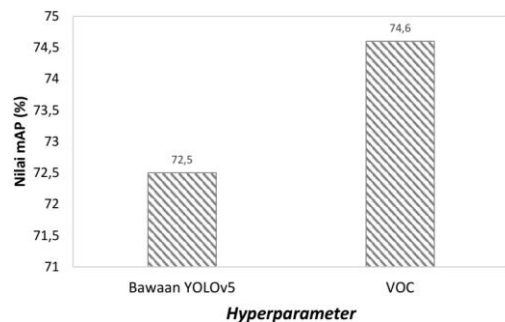
Visualisasi grafik *loss* pada model dengan *learn-ing rate* (a) 0,1; (b) 0,01; dan (c) 0,00334

Performa model dari konfigurasi nilai LR juga dianalisis dengan visualisasi grafik *loss* hasil *training* dan *validation*. Visualisasi grafik *loss* dibagi menjadi 3 bagian, yaitu *box loss*, *classification loss*, dan *object loss*. Gambar 11 menunjukkan visualisasi grafik *loss* pada setiap model. Pada model dengan nilai LR tertinggi, dapat dilihat bahwa di awal *epoch* terdapat *noise* yang cukup tinggi pada setiap grafik *loss*. *Noise* pada kurva *loss* menunjukkan ketidakstabilan, ditandai dengan nilai *loss* yang bergerak naik turun. Dengan LR 0,01, *noise* yang

terlihat pada grafik *loss* terdapat pada kurva *validation class loss* dan *object loss* saja. Sedangkan, model terakhir menghasilkan grafik *loss* yang paling mulus dengan tingkat *noise* yang relatif rendah. Analisis grafik *loss* membuktikan bahwa nilai LR yang tinggi menghasilkan koreksi bobot yang besar setiap model meng-*update* parameter. Hal ini mengakibatkan hasil yang terlalu menyimpang dari nilai optimal dan terjadi ketidakstabilan. Hal tersebut dapat dihindari dengan menggunakan nilai LR yang lebih rendah, sehingga jalannya proses pembelajaran untuk mencapai lokal minimum lebih stabil.

C. Konfigurasi Hyperparameter pada Model YOLOv5

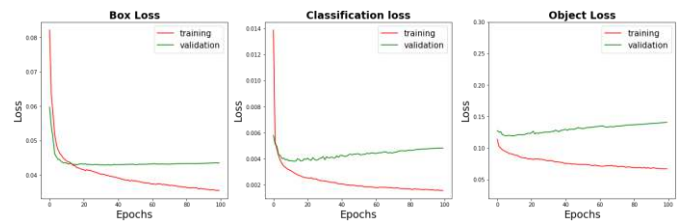
Percobaan optimasi model YOLOv5 dilakukan dengan dua masukan *hyperparameter*, yaitu VOC dan bawaan YOLOv5. Nilai *learning rate* terbaik yang didapat dari percobaan sebelumnya (0,00334) akan digunakan untuk masukan pada pelatihan model di percobaan ini. Hasil mAP terbaik dari percobaan ini dicapai oleh *hyperparameter* VOC, dengan nilai 74,6%. Perbandingan mAP terhadap *hyperparameter* VOC dan bawaan YOLOv5 dapat dilihat pada Gambar 12.



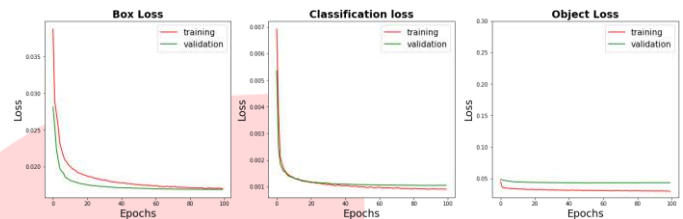
GAMBAR 12

(Grafik perbandingan nilai mAP konfigurasi *hyperparameter*)

Pada gambar tersebut, dapat diketahui bahwa model yang menggunakan *hyperparameter* VOC unggul berdasarkan nilai mAP, dengan selisih 2,1%. Selain nilai mAP yang meningkat, penulis menyadari adanya perubahan grafik *loss* yang secara signifikan semakin baik ketika menggunakan *hyperparameter* VOC. Gambar 13 menunjukkan perbandingan grafik *loss* dari hasil *training* model dengan masing-masing *hyperparameter*. Model dengan *hyperparameter* bawaan YOLOv5 terindikasi mengalami *overfitting*. Hal tersebut dapat dilihat dari kurva *validation loss* yang bergerak naik atau menjauhi nilai minimum, dibandingkan dengan kurva *training loss* yang bergerak sebaliknya. Sedangkan, model yang dilatih dengan *hyperparameter* VOC menunjukkan hasil yang baik (*good fit*), karena jarak antara kurva *training loss* dan *validation loss* yang berdekatan. Pengaruh dari penggunaan *hyperparameter* VOC terhadap peningkatan performa model YOLOv5 dapat diakibatkan karena setiap *dataset* cenderung memiliki perbedaan dalam pemilihan nilai-nilai *hyperparameter*. *Hyperparameter* setiap model memiliki nilai optimal yang berbeda untuk mencapai performa terbaik terhadap *dataset* yang berbeda pula. Pada kasus ini, hasil menunjukkan bahwa performa model YOLOv5 yang dilatih dengan *dataset* Dehazed Foggy



(a)



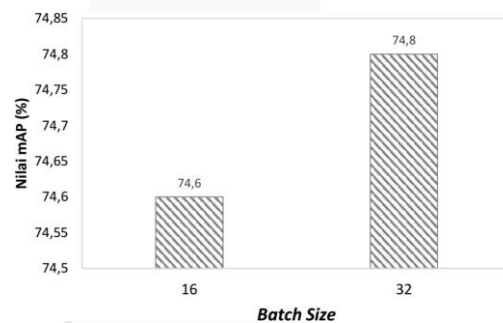
(b)

GAMBAR 13

(Visualisasi grafik *loss* pada model dengan *hyperparameter* (a) bawaan YOLOv5 dan (b) VOC)

Citiescapes lebih baik ketika menggunakan *hyperparameter* VOC.

D. Konfigurasi Batch Size pada Model YOLOv5



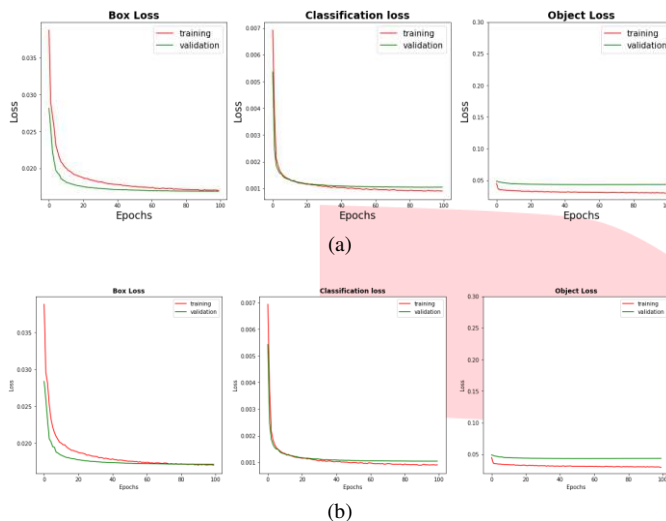
GAMBAR 14

(Grafik perbandingan nilai mAP pada konfigurasi *batch size*)

Percobaan optimasi terakhir pada model YOLOv5 dilakukan dengan membandingkan pelatihan model dengan dua masukan nilai *batch size*, yaitu 16 dan 32. Sedangkan, masukan *learning rate* dan *hyperparameter* lainnya diambil dari hasil terbaik dari percobaan sebelumnya. Hasil mAP tertinggi bernilai 74,8% dicapai oleh model yang dilatih dengan *batch size* 32. Perbandingan mAP terhadap *batch size* dapat dilihat pada Gambar 14.

Percobaan ini membuktikan bahwa model dengan ukuran *batch size* yang lebih tinggi akan menghasilkan akurasi yang lebih tinggi pula. Dalam hal ini, model dengan *batch size* 32 unggul dengan selisih mAP sebesar 0,2%. *Batch size* menentukan jumlah sampel yang akan dilatih pada jaringan dalam satu *epoch*. Semakin kecil nilai *batch size*, pengoreksian bobot pada jaringan akan lebih sering dilakukan tanpa mendapatkan informasi dari *dataset* secara utuh. Hal ini menyebabkan

tingkat akurasi model menjadi lebih rendah dibandingkan menggunakan *batch size* yang besar karena perkiraan model akan semakin akurat dan kinerja model dalam penyesuaian bobot akan meningkat.



GAMBAR 15

(Visualisasi grafik *loss* pada model dengan *batch size* (a) 16 (b) 32)

Visualisasi grafik *loss* yang dihasilkan kedua model dapat dilihat pada Gambar 15. Ketika dibandingkan antara grafik *loss* yang dihasilkan model dengan *batch size* 16 dan 32, kedua grafik tersebut tidak memiliki perbedaan yang signifikan. Jika diteliti, grafik *box loss* yang dihasilkan model pertama memiliki kurva *training loss* dan *validation loss* yang saling berhimpit pada epoch 80-100. Sedangkan, pada model kedua, *training loss* dan *validation loss* mulai saling berhimpit dari epoch ke-60. Hal ini menunjukkan bahwa model dengan ukuran *batch* 32 dapat mencapai nilai optimal lebih awal dibandingkan dengan model pertama. Maka, penggunaan ukuran *batch* yang besar akan membuat jumlah epoch yang dibutuhkan ketika *training* menjadi lebih sedikit.

V. KESIMPULAN

Konfigurasi model *Collaborative Learning* berdasarkan *Two-Branch Neural Network* sebagai *image dehazing sub-network* dan YOLOv5 sebagai *object detection sub-network* telah berhasil dilakukan pengujian dan analisis. Modifikasi *hyperparameter* pada model YOLOv5 juga berhasil diimplementasikan. Model *Collaborative Learning* mendapatkan performansi terbaik berdasarkan nilai mAP dibandingkan dengan YOLOv5 orisinal dengan nilai 71,5%. Di sisi lain, terjadi peningkatan performansi berdasarkan nilai mAP dan visualisasi grafik *loss* setelah proses *hyperparameter tuning* pada YOLOv5 dilakukan. Adapun beberapa kesimpulan dari hasil pengujian dan analisis adalah sebagai berikut:

- 1) Konfigurasi model *Collaborative Learning* memiliki mAP yang lebih tinggi dalam mendeteksi objek untuk

Autonomous Vehicle pada lingkungan berkabut dibandingkan YOLOv5 orisinal.

- 2) Terjadi peningkatan performansi setelah dilakukan *hyperparameter tuning* dengan konfigurasi terbaik yaitu *learning rate* 0,0034; *hyperparameter* VOC, dan *batch size* 32.
- 3) Berdasarkan hasil grafik *loss*, *learning rate* memiliki pengaruh pada nilai *loss* yang stabil dan penggunaan *hyperparameter* VOC memperbaiki *overfitting* pada model YOLOv5 yang dilatih dengan *dataset* Foggy Cityscapes.

REFERENSI

- [1] F. Munir, S. Azam, M. I. Hussain, A. M. Sheri, and M. Jeon, "Autonomous vehicle," *Proceedings of the 2018 International Conference on Sensors, Signal and Image Processing - SSIP 2018*, 2018.
- [2] S. Zang, M. Ding, D. Smith, P. Tyler, T. Rakotoarivelo, and M. A. Kaafar, "The impact of adverse weather conditions on autonomous vehicles: How rain, snow, fog, and hail affect the performance of a self-driving car," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, p. 103–111, 2019.
- [3] B. Li, X. Peng, Z. Wang, J. Xu, and D. Feng, "Aod-net: All-in-one dehazing network," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [4] Y. Yu, H. Liu, M. Fu, J. Chen, X. Wang, and K. Wang, "A two-branch neural network for non-homogeneous dehazing via ensemble learning," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.
- [5] S.-C. Huang, T.-H. Le, and D.-W. Jaw, "Dsnnet: Joint semantic learning for object detection in inclement weather conditions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2020.
- [6] G. Jocher, A. Stoken, J. Borovec, NanoCode012, A. Chaurasia, TaoXie, L. Changyu, A. V. Laughing, Tkianai, and et al., "ultralytics/yolov5: v5.0 - yolov5-p6 1280 models, aws, supervise.ly and youtube integrations," Apr 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.4679653>
- [7] C. Sakaridis, D. Dai, and L. V. Gool, "Semantic foggy scene understanding with synthetic data," *International Journal of Computer Vision*, vol. 126, no. 9, p. 973–992, 2018.
- [8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [9] M. Kalinina and P. Nikolaev, "Research of yolo architecture models in book detection," *Proceedings of the 8th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2020)*, 2020.
- [10] D. Thuan, "Evolution of yolo algorithm and yolov5: The state-of-the-art object detection algorithm," 2021.
- [11] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: <https://arxiv.org/abs/2004.10934>
- [12] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-

- H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.
- [13] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.
- [14] G. I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, and H. Samulowitz, "Aneffective algorithm for hyperparameter optimization of neural networks," *IBM Journal of Research and Development*, vol. 61, no. 4/5, pp. 9–1, 2017.
- [15] D. R. Wilson and T. R. Martinez, "The need for small learning rates on large problems," in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, vol. 1. IEEE, 2001, pp.115–119.
- [16] B. D. Hammel, "What learning rate should i use?" Mar 2019. [Online]. Available: <http://www.bdhammel.com/learning-rates/>
- [17] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," 2017.
- [18] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43,no. 2, p. 652–662, 2021.
- [19] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Be- nenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.