Jurnal Ismetek *ISSN 2406-9841*
Vol. 21 No.02 Juni Tahun 2026 *e-IISN 2986-2973*

PENERAPAN *CONVOLUTIONAL NEURAL NETWORK* (CNN) UNTUK DETEKSI DAN ESTIMASI JARAK KENDARAAN

Lola
Program Studi Teknik Informatika, FTI, Institut Teknologi Budi Utomo Jakarta,
lola.rezak@gmail.com

Abstrak
Deteksi dan estimasi jarak kendaraan adalah dua tugas yang perlu dilakukan oleh kendaraan otonom. Perkembangan pembelajaran mendalam (*deep learning*), khususnya CNN, memungkinkan pemrosesan citra untuk berbagai tugas. Pada makalah ini, dibahas penggunaan CNN untuk deteksi dan estimasi jarak kendaraan dengan model Dist-YOLO. Model dibangun dengan *backbone* Xception dan *head* YOLOv3 yang dimodifikasi serta dilatih menggunakan *dataset* KITTI. Berdasarkan percobaan yang dilakukan, model sudah dapat mendeteksi dan mengestimasi jarak kendaraan dengan cukup baik. Peningkatan lebih lanjut yang dapat dilakukan adalah optimasi model, sehingga dapat melakukan inferensi secara *real-time*.

Kata Kunci : Deteksi Kendaraan, Estimasi Jarak Kendaraan, CNN, YOLO, Dist-YOLO

1. PENDAHULUAN

Mendeteksi kendaraan yang ada dan memperkirakan jaraknya adalah dua tugas yang perlu dilakukan kendaraan otonom. Terdapat dua jenis sensor yang dapat digunakan untuk estimasi jarak, yaitu sensor aktif (misalnya lidar dan radar) dan sensor pasif (misalnya kamera). Estimasi jarak sensor aktif lebih akurat dan algoritma pemrosesan jaraknya lebih sederhana, tetapi harganya lebih mahal daripada sensor pasif (J.Brummelen, 2021).

Ide yang mendasari estimasi jarak dengan kamera adalah kemampuan manusia dalam tugas tersebut dengan hanya masukan mata. Otak manusia mampu memperkirakan posisi objek-objek dengan hanya melihatnya. Teknologi kecerdasan buatan dan pembelajaran mendalam (*deep learning*) memungkinkan komputer untuk melakukan tugas tersebut.

Deteksi kendaraan dengan pembelajaran mendalam dapat dilakukan dengan dua pendekatan, yaitu satu tahap dan dua tahap. Pendekatan satu tahap memiliki waktu pemrosesan yang lebih cepat, tetapi kinerjanya mungkin tidak seakurat pendekatan dua tahap (L. Liu, 2023). (Redmon, 2021) mengusulkan model satu tahap berbasis CNN (*convolutional neural network*) yang bernama YOLO (*You Only Look Once*) untuk tugas deteksi. Model tersebut memiliki kinerja yang baik dan waktu pemrosesan yang cepat, sehingga cocok untuk berbagai aplikasi yang membutuhkan deteksi objek.

Pada makalah ini, dibahas tugas deteksi objek dan estimasi jarak kendaraan menggunakan CNN. Estimasi jarak dapat dilakukan secara langsung dalam model deteksi objek atau secara terpisah berdasarkan hasil deteksi objek. Model Dist-YOLO (Vajgl, 2022) digunakan karena mampu mengestimasi jarak langsung pada vektor hasil prediksi model.

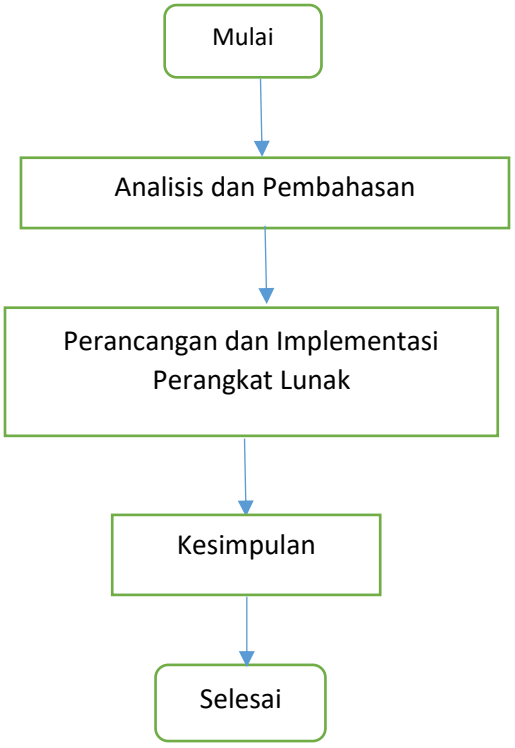
Implementasi model dibuat dengan model *backbone* pra-latih Xception dan pelatihan dilakukan pada *dataset* KITTI yang memiliki label jarak. Terakhir, eksperimen pada berbagai citra masukan dilakukan dan hasilnya dievaluasi.

Gambar 1. Berbagai Sensor pada Kendaraan Otonom
Sumber : Vajgl, 2022

ISSN 2406-9841
e-IISN 2986-2973

2. METODOLOGI

Metodologi penelitian digambarkan dalam bentuk diagram alir sebagai berikut:



Gambar 2. Diagram Alir Metodologi Penelitian
Sumber : https://www.researchgate.net/publication/338235695_Metode-Metode_Penelitian_Dalam_Penulisan_Jurnal_Ilmiah_Elektronik

3. HASIL DAN PEMBAHASAN

3.1 HASIL

Model diujicobakan pada beberapa contoh citra yang memuat satu atau lebih kendaraan. Waktu inferensi model diukur pada mesin dengan Nvidia T4 dan didapatkan waktu pemrosesan yang tertera pada gambar 3. Model memerlukan waktu inferensi di atas 1 detik, sehingga belum mampu melakukan deteksi dan estimasi jarak secara *real-time*.

Gambar 3. Waktu Inferensi Satu Citra
Sumber : Penelitian mandiri

Tabel 1 memperlihatkan beberapa contoh hasil deteksi dan estimasi jarak kendaraan pada beberapa citra. Terdapat keterangan beberapa estimasi jarak di bawah citra, dari kendaraan paling depan ke paling belakang, karena ukuran label yang mungkin terlalu kecil untuk dilihat di laporan. Semua satuan jarak adalah dalam meter. Empat baris pertama adalah citra dari *dataset* KITTI, sedangkan empat baris berikutnya adalah citra jalan di Kota Bandung (Ulin, 2025).

Model sudah cukup baik melakukan prediksi, bahkan pada citra yang cukup berbeda dengan data latih, yaitu citra di Kota Bandung. Model dapat mendeteksi kendaraan-kendaraan pada citra dan memperkirakan kedalamannya (jarak dari kamera). Akan tetapi, kinerjanya masih dapat ditingkatkan lagi. Kendaraan yang jauh nampak kecil, sehingga beberapa masih belum terdeteksi (misal baris 8). Kendaraan dengan model yang cukup berbeda juga beberapa masih belum terdeteksi (misal baris 7). Model dapat dilatih lagi pada beragam *dataset* dan berbagai tempat sehingga kemampuan deteksi dan estimasinya dapat lebih baik.

Tabel 1. Beberapa Contoh Hasil Deteksi dan Estimasi Jarak

No	Prediksi
1	 (Keterangan: 10.5, 16.6, 19.5, ...)
2	 (Keterangan: 25.2, 34.5, 40.3, 77.3, ...)
3	 (Keterangan: 3.5, 11.2, 16.6, 22.9, ...)
4	 (Keterangan : 6.8, 12.1, 19.6, …)
5	 (Keterangan: 17.9, 25, 28.6, ...)
6	 (Keterangan: 8.9, 6, 12.7, ...)
7	 (Keterangan: 29.9)
8	 (Keterangan: 12.5, 19.7)

Sumber : Dokumentasi penelitian mandiri

3.2 PEMBAHASAN

A. Pembelajaran Mendalam

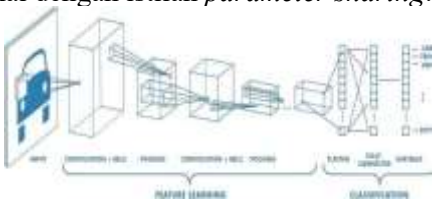
Pembelajaran mendalam (*deep learning*) adalah pembelajaran mesin yang menggunakan jaringan saraf buatan dengan tiga atau lebih lapisan. Jaringan saraf tersebut meniru cara kerja otak manusia dan memungkinkan pembelajaran pada data yang besar. Pembelajaran mendalam ada dalam beragam produk dan layanan sehari-hari, misalnya asisten digital, deteksi *fraud* kartu kredit, hingga *self-driving cars*.

Pembelajaran mendalam adalah *subset* dari pembelajaran mesin. Pembedanya dengan pembelajaran mesin klasik adalah tipe data dan metode pembelajaran yang digunakan. Pembelajaran mendalam mampu menghilangkan beberapa tahap pra-proses data dan memproses data yang tidak terstruktur, misalnya teks dan citra.

B. Jaringan Saraf Konvolusi

Jaringan saraf konvolusi (*convolutional neural network* / CNN) adalah jaringan saraf buatan yang memiliki kinerja baik pada tugas yang menerima data masukan berupa citra, suara, atau sinyal suara. CNN memiliki tiga tipe utama lapisan, yaitu lapisan konvolusi, lapisan *pooling*, dan lapisan *fully-connected* (FC).

Lapisan konvolusi adalah komponen utama dalam CNN, tempat kebanyakan komputasi terjadi. Lapisan ini membutuhkan beberapa komponen, yaitu data masukan, filter, dan peta fitur. Filter adalah *array* bobot yang akan dikonvolusikan dengan data masukan. Nilai bobot dalam filter tetap ketika dioperasikan pada berbagai bagian citra, dikenal dengan istilah *parameter sharing*.



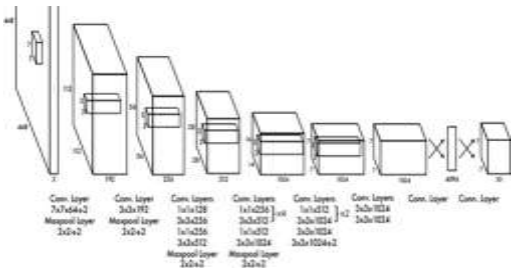
Gambar 4. Berbagai Lapisan pada CNN
Sumber : https://www.towardsdatascience.com

Lapisan *pooling* melakukan *downsampling* atau pengurangan dimensi, sehingga mengurangi banyaknya parameter masukan. Operasi pada lapisan ini mirip dengan lapisan konvolusi, tetapi filter yang digunakan bukanlah bobot, melainkan fungsi agregat. Terdapat dua tipe *pooling*, yaitu *max pooling* dan *average pooling*.

Lapisan *fully-connected* (FC) adalah lapisan yang melakukan tugas klasifikasi berdasarkan fitur yang telah diekstraksi pada berbagai lapisan sebelumnya. Pada lapisan ini, tiap *neuron* terhubung dengan seluruh *neuron* pada lapisan sebelumnya. Lapisan ini biasanya menggunakan fungsi aktivasi softmax untuk mengklasifikasikan masukan.

C. *You Only Look Once* (YOLO)

YOLO (Redmon, 2021) adalah model CNN yang mampu melakukan deteksi objek dengan kinerja yang baik dan waktu eksekusi yang cepat. YOLO menerapkan pendekatan satu tahap, yaitu langsung memprediksi kotak pembatas objek dalam sekali langkah. Arsitektur YOLO dapat dilihat pada gambar 5.

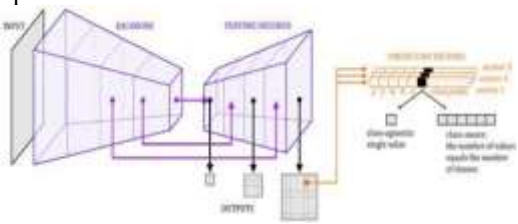


Gambar 5. Arsitektur YOLO
Sumber : Redmon, 2021

Pelatihan model ini terdiri atas dua tahap, yaitu *pre-training* dan *fine-tuning*. Pada tahap *pre-training*, 20 lapisan konvolusi pertama dilatih pada ImageNet secara *supervised*. Pada tahap *fine-tuning*, model dilatih pada tugas deteksi dengan tambahan 4 lapisan konvolusi dan 2 lapisan *fully-connected*. Hasil prediksi pada lapisan terakhir adalah probabilitas kelas objek dan kotak pembatas (*bounding box*) objek tersebut.

D. Dist-YOLO

Dist-YOLO (Vajgl, 2022) adalah varian YOLO yang mampu memprediksi jarak absolut objek hanya dengan informasi dari sebuah citra kamera monokuler. Model tersebut memberikan estimasi jarak dengan galat relatif 11% pada pengujian di *dataset* KITTI. Modifikasi yang dilakukan juga tidak berpengaruh besar pada waktu eksekusi jika dibandingkan dengan model YOLO tanpa modifikasi.



Gambar 6. Arsitektur Dist-YOLO
Sumber : Vajgl, 2022

Dist-YOLO terdiri atas dua bagian, yaitu *backbone* dan *feature decoder* (*head*). *Backbone* bertindak sebagai *feature extractor* dan menggunakan model pra-latih, misalnya Darknet-53 dan Xception. *Head* adalah kepala YOLO yang dimodifikasi dan memiliki tugas melakukan deteksi dan estimasi jarak. Modifikasi yang dilakukan pada kepala YOLO agar dapat melakukan estimasi jarak adalah perubahan vektor prediksi dan fungsi *loss* yang digunakan.

Deteksi dan estimasi jarak kendaraan menggunakan model Dist-YOLO (Vajgl, 2022). Terdapat dua bagian pada model ini, yaitu *backbone* dan *head*. *Backbone* adalah model pra-latih yang telah dilatih pada *dataset* citra yang besar, misalnya ImageNet. *Backbone* disambungkan ke *head* YOLO yang dimodifikasi vektor keluarannya untuk *fine-tuning* pada tugas deteksi dan estimasi jarak. Pada implementasi model, digunakan *backbone* Xception dan *head* YOLOv3 (dengan modifikasi). Pelatihan *fine-tune* dilakukan menggunakan *dataset* KITTI.

E. Pengumpulan *Dataset*

Dataset yang digunakan adalah KITTI (Geiger, 2023) yang telah diproses anotasi jaraknya (Vajgl, 2022). Anotasi (label) terdiri atas 6 nilai per objek, yaitu 4 nilai koordinat x dan y sudut kiri atas dan kanan bawah *bounding box*, 1 nilai kelas, dan 1 nilai jarak dalam meter. Terdapat 7 kelas yang digunakan, yaitu Pedestrian, Car, Van, Truk, Person_Sitting, Cyclist, dan Tram.

F. Pembuatan Model

Pembuatan model menggunakan beberapa pustaka yang disediakan oleh (Vajgl, 2022). Model dibuat dengan membuat sebuah model baru yang merupakan gabungan *backbone* Xception yang diperoleh dari pustaka Keras dan *head* YOLOv3 (dimodifikasi) yang disusun sendiri dalam kode. Berikut adalah kode untuk melakukan penyusunan tersebut.

```
out_filters = self.num_anchors * (self.num_classes + 3+1)
                    backbone = Xception(
                    input_tensor=input_tensor,
                    weights='imagenet', include_top=False)
backbone_len = 132

f1 = backbone.get_layer('block14_sepconv2_act').output
f2 = backbone.get_layer('block13_sepconv2_bn').output
f3 = backbone.get_layer('block4_sepconv2_bn').output
f1_channel_num = 1024
f2_channel_num = 512
f3_channel_num = 256

x, y1 = make_last_layers(f1, f1_channel_num//2, out_filters,
predict_id='1')
x = compose(
    DarknetConv2D_BN_Leaky(f2_channel_num//2, (1,1)),
    UpSampling2D(2))(x)
x = Concatenate()([x,f2])
x, y2 = make_last_layers(x, f2_channel_num//2, out_filters,
predict_id='2')
x = compose(
    DarknetConv2D_BN_Leaky(f3_channel_num//2, (1,1)),
    UpSampling2D(2))(x)
x = Concatenate()([x, f3])
x, y3 = make_last_layers(x, f3_channel_num//2, out_filters,
predict_id='3')
```

Gambar 7. Kode Penyusunan Model Gabungan *Backbone* Xception
Sumber : Vajgl, 2022

Sebagaimana dapat diperhatikan pada kode, Dist-YOLO dibuat dengan menggabungkan *backbone* dan *head*. Perbedaan YOLOv3 dan Dist-YOLO adalah dimensi vektor keluaran dan fungsi *loss* yang digunakan. Pada kode diatas yang merupakan Dist-YOLO, ukuran *out_filter* adalah jumlah *anchor* yang digunakan dikalikan dengan banyak kelas + 6. Sedangkan pada YOLOv3, ukurannya jumlah *anchor* dikalikan banyak kelas + 5. Perbedaan ini karena pada Dist-YOLO terdapat data tambahan jarak yang perlu diprediksi. Perbedaan lainnya adalah fungsi *loss* pada Dist-YOLO dimodifikasi sehingga mempertimbangkan data jarak.

C. Pelatihan

Pelatihan dilakukan menggunakan citra sejumlah 1000, dibagi menjadi 800 data latih dan 100 data validasi. Pelatihan dilakukan sebanyak 50 *epoch* dengan ukuran *batch* 16. Hasil pelatihan dapat disimpan untuk kemudian digunakan dalam melakukan inferensi.

4. KESIMPULAN

Jaringan saraf konvolusi (CNN) dapat digunakan untuk berbagai tugas, antara lain untuk deteksi dan estimasi jarak kendaraan. Model deteksi satu tahap YOLO dapat dimodifikasi sehingga menghasilkan Dist-YOLO, yaitu model yang mampu mendeteksi dan mengestimasi jarak kendaraan. Berdasarkan percobaan yang dilakukan berbasis model telah cukup baik dalam membedakan jarak atau kedalaman kendaraan dari kamera, meskipun masih terdapat galat dan belum mampu memproses secara *real-time*.

DAFTAR PUSTAKA

1. Geiger, A, Lenz, P, Stiller, C, Urtasun, R., (2023), “Vision meets robotics: The KITTI dataset,” Int J Rob Res, vol. 32, no. 11, 1231–1237.
2. J. Brummelen, Van, M. O’Brien, D. Gruyer, Najjaran, H., (2021), “Autonomous vehicle perception: The technology of today and tomorrow”, Transp Res Part C Emerg Technol, vol. 89, 384–406.
3. L. Liu., (2023), “Deep Learning for Generic Object Detection: A Survey”, Int J Comput Vis, vol. 128, no. 2, 261–318

4. Redmon, J, S. Divvala, R. Girshick, Farhadi, A., (2021), "You Only Look Once: Unified, Real-Time Object Detection ", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 779–788.
5. Vajgl, M, Hurtik, P, Nejezchleba, T., (2022), "Dist-YOLO: Fast Object Detection with Distance Estimation", Applied Sciences, vol. 12, no. 3, 1354.
6. Ulin, Opas., (2025), "KELILING KOTA BANDUNG || Suasana kota Bandung di pagi hari," https://youtu.be/rs5OX4JpnMc.