FINAL REPORT
INDONESIAN SIGN LANGUAGE (BISINDO) DETECTION

Oleh :

Kelompok 4

1. Sabrina Adinda Sari (1301194183)
2. Jane Raihan (1301194240)
3. M. Ridha Anshari (1301194071)

Abstract

Bahasa isyarat merupakan sistem komunikasi yang terdiri dari seperangkat simbol atau karakter tertulis untuk berbicara yang digunakan oleh tuna rungu untuk berkomunikasi dengan orang lain. Dalam penelitian ini, kami menggunakan metode YOLO untuk membangun citra model deteksi objek. Dataset yang kami gunakan adalah dataset Bahasa Isyarat Indonesia (BISINDO) yang kami peroleh dari Kaggle, dataset BISINDO sendiri berisi 390 citra yang terdiri dari 26 kelas (A-Z), dimana masing-masing kelas terdapat 15 citra. Pada hasil eksperimen yang dilakukan menampilkan mAP *score* sebesar 86.8% pada percobaan pertama dan 80.5% pada percobaan kedua.

Kata kunci - YOLO, Bahasa Isyarat Indonesia, BISINDO, Deteksi Objek, mAP.

A. Introduction

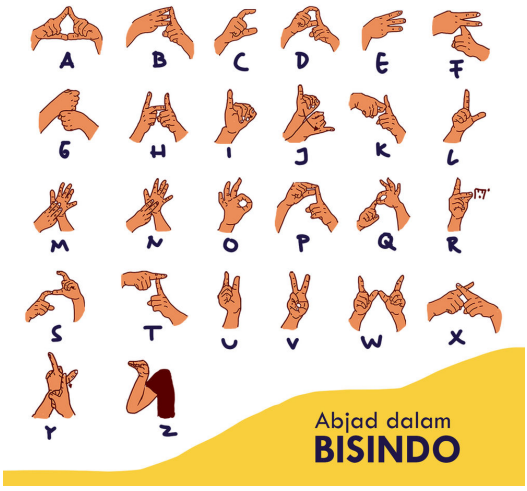

Figure 1 : Alphabet Bisindo

Pada *final project* kali ini kami akan membuat deteksi objek untuk Bahasa Isyarat Indonesia (BISINDO). Bisindo dibentuk oleh komunikasi Tuli dan muncul secara alami berdasarkan pengamatan komunitas Tuli Indonesia. Bisindo disampaikan dengan gerakan dua

tangan, dan lebih mudah dipahami oleh teman tuli. Alasan kami mengangkat tema ini adalah karena kami ingin membantu orang Tunarungu untuk berkomunikasi di ruang publik dan dalam kehidupan sehari-hari mereka, maka dari itu kami memutuskan untuk mengangkat tema Deteksi Objek untuk Bahasa Isyarat Indonesia (Bisindo). Bisindo sendiri di peragakan dengan menggunakan dua tangan, namun ada juga yang hanya menggunakan satu tangan seperti abjad : C, E, I, L, O, R, U, V, dan Z, selain itu di peragakan dengan menggunakan dua tangan.

B. Related Works

Penelitian deteksi objek untuk BISINDO sendiri sudah banyak dilakukan oleh kebanyakan orang, namun metode yang mereka gunakan rata-rata adalah menggunakan *machine learning*, R-CNN (*Region Based Convolutional Neural Networks*), dan SSD (*Single Shot Detector*). Masih jarang ditemukan penelitian deteksi Bisindo dengan menggunakan metode YOLO.

YOLO merupakan metode yang banyak digunakan orang untuk deteksi objek baru-baru ini, namun tidak untuk dataset Bisindo. YOLO sendiri dikatakan sebagai metode objek deteksi yang bagus dibandingkan dengan metode-metode terdahulunya. Yolo v.4 merupakan versi yang paling baik ketika digunakan untuk deteksi objek.

C. Data

Untuk *dataset* BISINDO yang digunakan diambil dari kaggle dengan total 390 citra yang terdiri dari 26 kelas (A-Z), dimana masing-masing kelas terdapat 10 citra. Setelah di download, maka dilakukan *preprocessing* citra dengan mengubah ukuran nya menjadi 416x416. Kemudian *dataset* tersebut dibagi menjadi *training* dan *testing*, untuk testing memakai 10% *dataset* total, lalu sisanya dipakai untuk *training*.

Figure 2 : Dataset

D. Methods

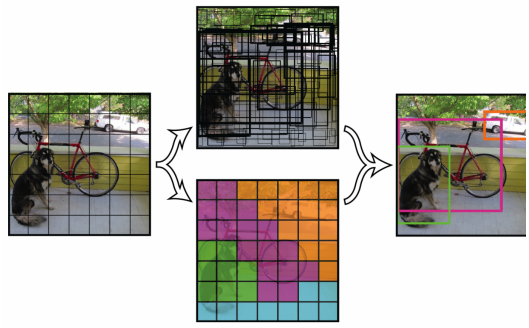

Figure 3 : YOLO

Metode yang kami gunakan adalah YOLO v4, karena YOLO v4 merupakan model *realtime* tercepat dan akurat untuk mendeteksi objek dibandingkan dengan versi YOLO yang sudah ada. Berikut adalah alur kerja dalam implementasi sistem deteksi bisindo.

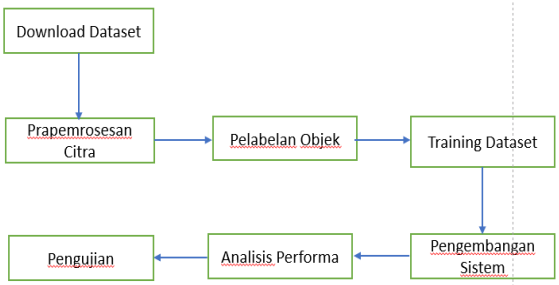

Figure 4 : Alur Kerja Sistem

Pertama, download dataset yang sudah tersedia di kaggle. Kedua lakukan pra-pemrosesan citra yaitu dengan melakukan *resize* citra yang awalnya berukuran 640x480 piksel menjadi 416x416 piksel. Ketiga lakukan anotasi pelabelan objek yang terdiri dari 26 kelas (A sampai Z), dimana satu citra kita set hanya terdapat satu objek saja. Disini kami melakukan anotasi menggunakan aplikasi web yang bernama Roboflow(https://roboflow.com/).

Keempat lakukan *training* dataset yang bertujuan untuk melatih komputer dengan cara mengolah citra dan anotasi yang telah dibuat sehingga terbentuk pola atau karakteristik dari masing-masing kelas yang akan menjadi bahan pertimbangan komputer dalam mencapai sebuah keputusan atau prediksi. Proses training dilakukan menggunakan *framework neural network Darknet* dengan konfigurasi.

Kelima akan dilakukan pengembangan sistem yang terdiri dari proses input citra dengan *network size* sebesar 416 × 416. Kemudian dilakukan *feature extraction* dengan pada YOLOv4

menggunakan Darknet53. YOLOv4 memberikan hasil prediksi berupa 6 nilai yang terdiri dari titik koordinat *bounding box* (tx, ty, tw, th), confidence dan class probability. Selain itu, YOLOv4 memberi prediksi *bounding box* dengan 3 skala yang berbeda. Masing-masing skala memiliki 3 *anchor* sehingga total memiliki 9 *anchor*.

Keenam, analisis performa memanfaatkan data *ground-truth box* yang merupakan dataset test yang telah diprediksi dengan cara memberi anotasi secara manual dan data *predicted box* yang merupakan data anotasi hasil dari sistem deteksi. Kemudian dua data tersebut diolah pada *Confusion Matrix* untuk dikelompokkan berdasarkan data *True Positive* (TF), *False Positive* (FP) dan *False Negative* (FN) yang sudah ditentukan batasannya pada *Intersect of Union* (IoU). Dari hasil pengelompokkan tersebut, kemudian dikalkulasi untuk mencari nilai *Precision, Recall, f1-score* dan *Average Precision* (AP) yang kemudian hasil akhirnya berupa nilai *mean Average Precision* (mAP).

Ketujuh lakukan pengujian terhadap dataset untuk testing dari kaggle dan kemudian dianalisis hasilnya termasuk ke dalam kelas mana serta menampilkan juga *evaluation metrics*-nya.

E. Experiments

Setelah dilakukan pra-pemrosesan dataset dan anotasi, kami melakukan *git clone* darknet. Darknet kami gunakan untuk konfigurasi.

```
1 !git clone https://github.com/AlexeyAB/darknet
```

Figure 5 : Install Darknet

Lalu kami melakukan update pada Makefile yang terletak pada folder darknet yang ada di drive yang sudah dilakukan *git clone*, setelah itu setting GPU = 1, CUDNN = 1, OPENCV = 1, kemudian *save* kembali setelah dilakukan konfigurasi ulang.

Setelah melakukan *update* pada makefile darknet, selanjutnya kami melakukan *fireplace* dataset ke darknet/data, setelah itu kami membuat file train.txt, test.txt, classes.names, image_data.data

Setelah selesai melakukan *fireplace* dataset, selanjutnya kami melakukan konfigurasi untuk training dan testing pada folder dataset/cfg. Untuk konfigurasi training, terlebih dahulu meng-*copy* isi konfigurasi yolov4.cfg sesuai template kemudian diubah bagian *value batch* = 32, *subdivisions* = 8, *max_batches* = 4000 (2000 * kelas), *steps* = 3200,3600, lalu pada bagian yolo *classes* = 2, pada bagian convolutional *filters* = 21, dan *mosaic* = 0. Untuk *classes* = 2, karena *fine tuning* untuk 2 kelas saja membutuhkan waktu bisa mencapai 5 jam lebih, apalagi jika full menggunakan 26 kelas, maka kami memilih untuk menggunakan kelas A dan B saja. Sehingga banyaknya filters=21, diperoleh dari kelas + 5(didapat dari file anotasi tiap image) * 3, yang kemudian diberi nama yolov4_train.cfg.

Selanjutnya untuk file yolov4_test.cfg hampir sama dengan konfigurasi yang ada pada yolov4_train.cfg hanya saja untuk value pada *batch* dan *subdivisions* yang berbeda, yaitu *batch* = 1, *subdivisions* = 1, *max_batches* = 4000 (2000 * kelas), *steps* = 3200,3600, lalu pada bagian *yolo classes* = 2, pada bagian *convolutional_filters* = 21, dan *mosaic* = 0.

Setelah dilakukan pengkonfigurasian pada masing-masing file yolov4_train.cfg dan yolov4_test.cfg, selanjutnya kami melakukan training model dengan dataset bisindo train.

```
os.chdir('/content/drive/My Drive/yolo-v4/darknet')
!sudo chmod +x darknet
!./darknet
```

Figure 6 : Training Model

Setelah dilakukan *training* model dengan dataset bisindo *train*, lalu dilakukan *training* model dengan dataset bisindo test.

Pada tahap ini kami melakukan 2 percobaan yaitu :

1) Percobaan ke-1 dengan menggunakan Learning Rate 0.01 dengan banyak iterasi training 1500. Dihasilkan grafik dan image training result seperti dibawah ini.

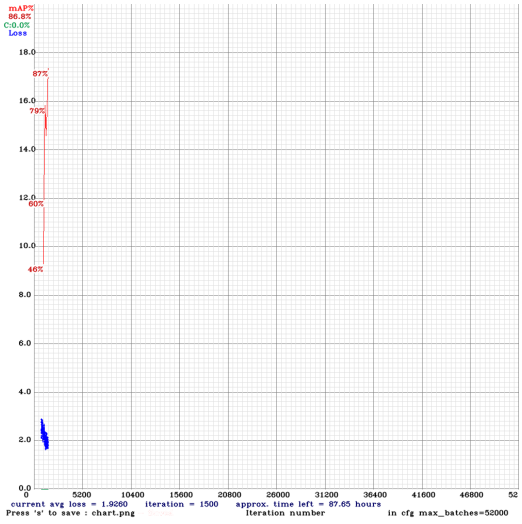

Figure 7 : Graph hasil modelling dengan LR 0.01

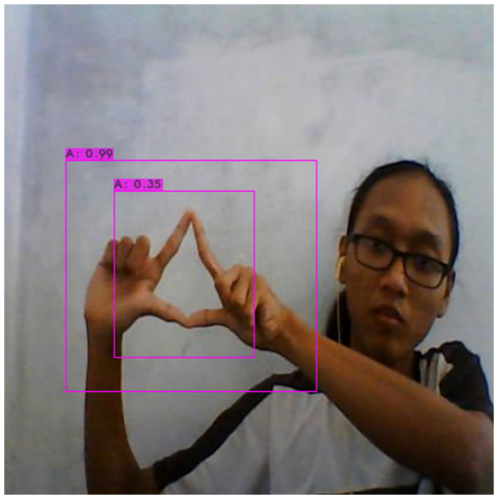

Figure 8 : Test Model

Image Testing Results	
for conf_thresh	25%
precision	60%
recall	92%
F1-score	73%
MAP	86.8%
TP	36
FP	24
FN	3
Average IoU	41.12%
IoU Treshold	50%
Total Detection Time	2 Second

Table 1: Image testing results

2) Percobaan ke-2 dengan menggunakan Learning Rate 0.0001 dengan banyak iterasi training 1500. Dihasilkan grafik dan image training result seperti dibawah ini.

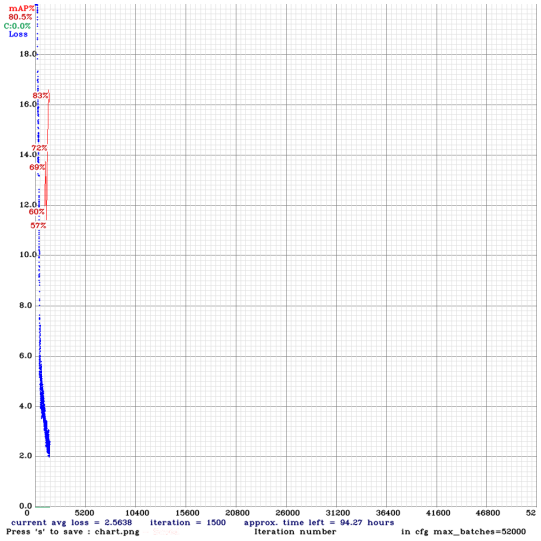

Figure 9 : Graph hasil modelling dengan LR 0.0001

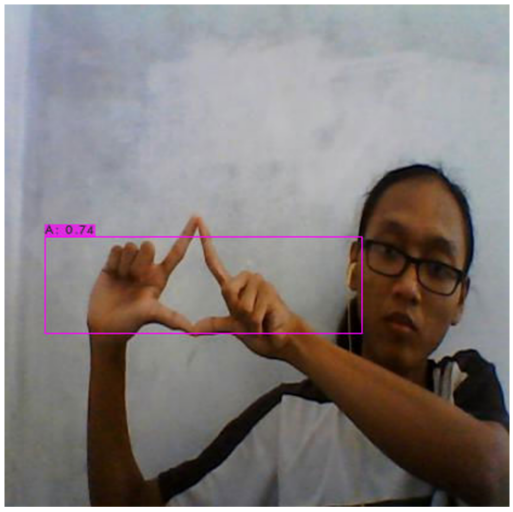

Figure 10 : Test Model

Image Testing Results	
for conf_thresh	25%
precision	58%
recall	72%
F1-score	64%
MAP	80.5%
TP	28
FP	20
FN	11
Average IoU	37.67%
IoU Treshold	50%
Total Detection Time	2 Second

Table 2: Image testing results

F. Conclusion

Dalam tugas besar ini kami membuat sistem deteksi Bisindo dengan menggunakan model Yolo-v4 dengan menggunakan 2 kelas yaitu kelas A dan B. Berdasarkan hasil percobaan pertama didapat mAP *score* sebesar 86.8% dengan *learning rate* 0.01. Untuk percobaan kedua didapatkan hasil mAP *score* sebesar 80.5 dengan *learning rate* 0.0001. Dapat disimpulkan bahwa *learning rate* sangat berpengaruh terhadap hasil dari mAP *score*.

Saran untuk kedepannya, kami akan mencoba mengembangkan sistem deteksi bisindo ini menjadi aplikasi dengan menggunakan semua kelas dari A sampai Z dan juga menggunakan dataset yang lebih banyak dan bervariasi.

G. References

[1] Hanafi, Y. U. (2020). *DETEKSI PENGGUNAAN HELM PADA PENGENDARA*. Surabaya.

[2] Olivia Kembuan, G. C. (2020). Convolutional Neural Network (CNN) for Image. 1-5.

[3] Steve Daniels, N. S. (2021). Indonesian Sign Language Recognition using YOLO Method. 6-7.

[4] Steve Daniels, N. S. (2021). Indonesian Sign Language Recognition using YOLO Method. 8.

[5] Xianlei Qiu, S. Z. (n.d.). Hand Detection For Grab-and-Go Groceries. 1.

[6]https://www.researchgate.net/publication/350082064_Indonesian_Sign_Language_Recognition_using_YOLO_Method

[7]https://youtube.com/playlist?list=PLlH6o4fAIji76vBRv54WPQr0MHgEiipL7