PENERAPAN YOLO DALAM KONVERSI PARTITUR LAGU DARI NOTASI BALOK MENJADI NOTASI ANGKA

ALVI ZAHRAN INDRASETYA[1], WINARNO SUGENG[1], THETA DINNARWATY PUTRI[1]

[1]Program Studi Teknik Informatika, Institut Teknologi Nasional Bandung
Email: alvizahran.azi@mhs.itenas.ac.id

Received 22 09 2021 **|** *Revised* 25 12 2021 **|** *Accepted* 25 12 2021

ABSTRAK

Notasi balok sering dijumpai pada partitur lagu, partitur lagu terbagi dua jenis diantaranya partitur notasi balok dan partitur notasi angka. Bagi sebagian pemusik pemula membaca notasi balok adalah hal yang sulit dibanding membaca notasi angka. Pada konversi notasi balok menjadi notasi angka secara manual kurang efektif dan efisien sehingga memerlukan sistem untuk lebih memudahkan dalam membaca notasi balok. YOLO (You Only Look Once) merupakan algoritma pengenalan objek dengan kemampuan pendeteksian secara cepat hanyamemproses gambar dengan sekali proses. Pada penelitian ini membangun sistem pengenalan notasi balok menggunakan algoritma YOLOv4 dengan backbone Darknet, tahapan terdiri dari dua bagian yaitu bagian data latih dan bagian data uji. Hasil yang diperoleh dari konversi ini memunculkan akurasi notasi yang sering muncul dalam partitur lagu Bingo notasi angka diperoleh pada notasi 1 do dengan akurasi 93.67% dan notasi yang sering muncul pada bagian chord akurasi diperoleh chord b sebesar 63.33%.

***Kata kunci**: Musik, Notasi, YOLO, Backbone Darknet*

ABSTRACT

Music notation is often found in sheet music, sheet scores are divided into two types, including score sheet notes and score notes for numbers. For somenovice musicians, reading music notation is more difficult than reading numerical notation. In converting music notation into numeric notation manually it is less effective and efficient, so it requires a system to make it easier to read music notation. YOLO (You Only Look Once) is an object recognition algorithm with fast detection capability, which only sees images in one process. In this study, building a music notation recognition system using the YOLOv4 algorithm with a Darknet backbone, the steps consist of two parts, namely the training data section and the test data section. The results obtained from this conversion bring up the accuracy of the notation that often appears in the score of the Bingo song. The numeric notation is obtained in the 1 do notation with an accuracy of 93.67% and the notation that often appears in the chord accuracy section is obtained by the chord b of 63.33%.

***Keywords**: Music, Notation, YOLO, Backbone Darknet*

1. PENDAHULUAN

Notasi balok merupakan sistem penulisan lagu atau sebuah karya musik yang dituangkan dengan bentuk sebuah notasi. Dalam notasi balok terbagi dari dua segmentasi *yaitu trebble clef* atau yang biasa ditulis dalam kunci G dan *bass clef* atau yang biasa ditulis dalam kunci F (Higgins, 2012). Pada umumnya *trebble clef* memainkan melodi musik sedangkan bass clef memainkan chord musik. Dalam notasi balok memiliki penulisan yang umum ditulis dalam partitur lagu. Notasi balok sering dijumpai pada partitur lagu, partitur lagu ini terbagi dua jenis diantaranya partitur notasi balok dan partitur notasi angka (Manus et al., 2010). Pada umumnya musisi harus mengerti partitur yang ditulis oleh notasi balok karena notasi balok adalah bentuk umum penulisan lagu atau musik yang lazim digunakan sejak abad pertengahan oleh Guido dari Arezzo (Reisenweaver, 2012). Dalam penulisan notasi balok yang sudah dituangkan kedalam partitur lagu ini memerlukan pengajar yang ahli di bidang musik.

Penerapan notasi balok ini memerlukan sistem untuk menyederhanakan pembacaan notasi, sebagai contohnya pada jurnal (Hajic & Pecina, 2017). Melakukan konversi untuk notasi yang ditulis dengan tulisan tangan dan menghasilkan notasi balok berbentuk MIDI, Penelitian yang berjudul 'Program Konversi Not Balok Dengan Struktur MusicXML Ke Not Angka' oleh (Chrisantyo A.A et al., 2007) melakukan penelitian untuk konversi notasi balok menjadi notasi angka dengan menggunakan MusicXML dan aplikasi Borland Delphi yang harus diunduh terlebih dahulu. Selanjutnya pada penelitian yang berjudul Pengembangan 'Aplikasi Konversi Representasi Not Balok Ke Not Angka Untuk Paduan Suara Campur' oleh(Chrisantyo et al., 2012) melakukan konversi notasi balok ke notasi angka dengan jenis partitur untuk paduan suara, dan pada penelitian yang berjudul 'Konversi Not Balok Menjadi Not Angka Dengan Metode OCR' oleh (I Wayan Aldon Alba Dona, 2021)melakukan konversinotasi balok menjadi notasi angka dengan penggunaan metode kendala dalam pendeteksian notasi balok OCR namun masih ditemukan beberapa. Dari beberapa penelitian diatas, penelitian ini akan dibagun sistem pengkonversian notasi balok menjadi notasi angka dengan menggunakan algoritma YOLO untuk memudahkan pemusik pemula membaca partitur musik.

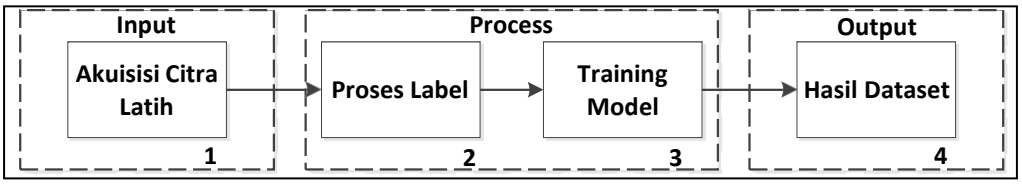
2. METODE PENELITIAN

Metode penelitian ini berisi tahapan perancangan yang terdiri dari perancangan sistem dan perancangan desain. Alur sistem penelitian ini digambarkan kedalam pemodelan *prototype*. Dasar dari penelitian ini menggunakan algoritma YOLO sebagai metode untuk pengenalan dan pendeteksian objek notasi balok. Penelitian ini bertujuan untuk mengimplementasikan algoritma YOLO kedalam sebuah pendeteksian gambar notasi balok dengan cara mengenali objek notasi balok secara satu- persatu, lalu setelah itu gambar notasi balok dilatih, selanjutnya notasi balok diuji dengan gambar notasi balok apakah dapat menghasilkan konversi notasi angka.

Dalam sistem untuk pengolahan citra digital terdapat 3 bagian utama pemrosesan yang terdiri dari *input, process*, dan *output*. Pada penelitian ini sebuah perancangan dibutuhkan untuk membangun sistem, perancangan ini akan digambarkan berupa diagram *block* dan data *flow diagram*.

2.1 *Block Diagram* Citra Latih

Pada diagram kali ini menjelaskan tentang gambaran citra latih mulai dari tahap akuisisi, pembuatan label, training model, dan hasil dataset.



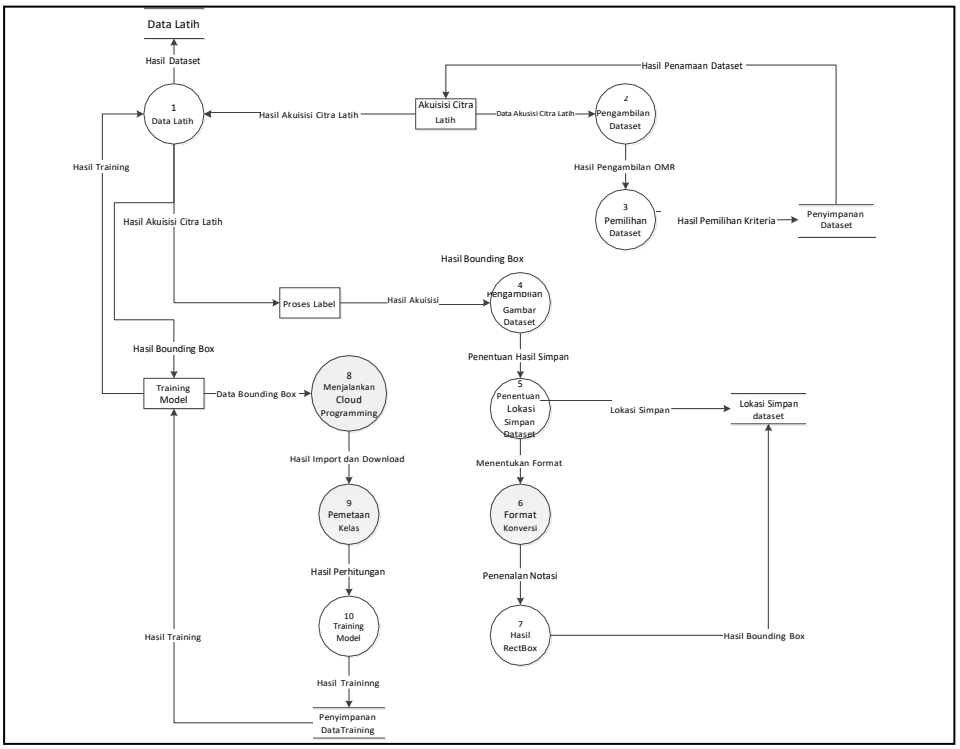
Gambar 1. *Block Digram* Citra Latih

Pada Gambar 1 merupakan *block diagram* dari citra latih, berikut penjelasan dari pointer blok diagram citra latih:

1. Akuisi citra latih merupakan tahapan input dari suatu sistem yang dibangun, citra notasi balok dimasukan ke dalam sistem setelah itu masuk ke tahap selanjutnya yaitu tahap pemrosesan.
2. Setelah citra diproses dengan penegnalan tiap notasi dengan memberikan label dan *bounding box*.
3. Setelah melalui proses label maka tahap selanjutnya melakukan pemrosesan training model. Dalam training model dilakukan beberapa tahapan diantaranya penggunaan algoritma YOLO sebagai proses dalam training model. Dari hasil proses *bounding box*, hasil dari pemrosesan ini berupa file format *.weights* sebagai hasil dari training dan hasildari training tersebut disimpan.

2.2 Data *Flow Diagram* Citra Latih

Pada data flow diagram citra latih ini menjelaskan bagaimana proses alur data dalam proses citra latih.

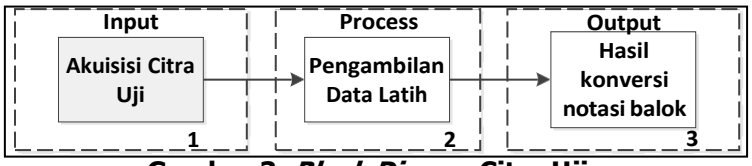


Gambar 2. DFD Citra Latih

Pada Gambar 2 merupakan alur DFD dari data citra latih yang bermulai dari data latih selanjutnya dari data latih mendapatkan data dari akuisisi citra. Didalam citra akuisisi citra latih ini melakukan pemrosesan untuk menghasilkan dataset yang sudah dibuat kriteria dan diberi format penamaan dan selanjutnya data tersebut diteruskan kedalam data latih. Dari data latih diteruskan kedalam proses label untuk dilakukan pengenalan notasi dan menghasilkan bounding box dan dari hasil *bounding box* ini di lanjutkan untuk melakukan *training* model untuk di lakukan proses training. Hasil dari proses ini disimpan kedalam penyimpanan data latih.

2.3 *Block Diagram* Citra Uji

Pada diagram kali ini menjelasakan gambaran citra uji pada sistem yang merupakan proses pendeteksian notasi balok yang meliputi akuisisi citra, pengambilan data latih, implementasi algoritma YOLO, dan hasil konversi.



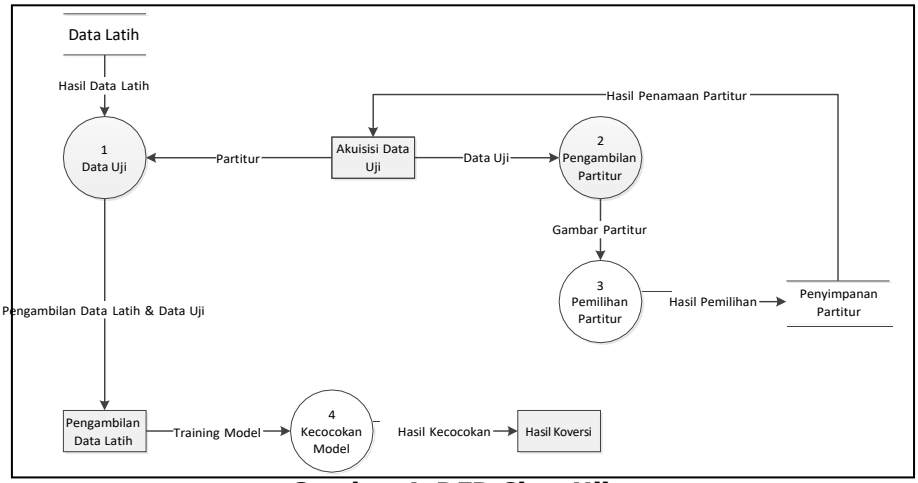
Gambar 3. *Block Digram* Citra Uji

Pada Gambar 3 merupakan blok diagram dari citra uji, berikut penjelasan dari pointer blok diagram citra uji:

1. Akuisi citra uji merupakan tahapan input dari suatu sistem yang dibangun, citra notasi balok dimasukan ke dalam sistem setelah itu masuk ke tahap selanjutnya yaitu tahap pemrosesan.
2. Setelah citra di masukan maka tahap selanjutnya melakukan pengambilan data latih.
3. Setelah melakukan tahap menggunakan algoritma YOLO. Dalam pemrosesan YOLO melakukan pencocokan hasil gambar yang telah melakukan training di cocokan dengan data uji yang dimasukan berupa partitur lagu. Tahap ini terdiri dari beberapa pemrosesan sepeti ekstraksi fitur dengan darknet setelah itu melakukan proses konvolusi, dan tahap terakhir melakukan prediksi *bounding box*.
4. Tahap terakhir dari blok diagram ini merupakan hasil dari konversi notasi.

2.4 *Data Flow Diagram* Citra Uji

Pada data flow diagram citra uji ini menjelaskan bagaimana proses alur data dalam proses citra uji.



Gambar 4. DFD Citra Uji

Pada Gambar 4 merupakan alur DFD dari data uji yang bermulai dari data uji selanjutnya data uji menerima hasil dari data latih yang sebelumnya sudah dilakukan pada sub-bab 2.3. Pada data uji menerima data partitur yang telah dilakukan pengambilan dan pemilihan dalam akuisisi citra uji. Dalam akuisisi citra uji ini mengelurkan hasil data berupa partitur yang sudah di berikan penamaan dan gambar partitur. Selanjutnya melakukan pengambilan data latih untuk diakukan pencarian kecocokan dan menghasilkan hasil konversi.

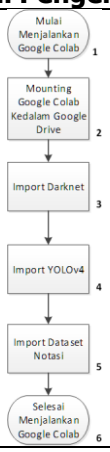
3. HASIL DAN PEMBAHASAN

Bagian ini menjelaskan bagaimana implementasi yang telah dilakukan serta hasil yang diperoleh. Implementasi dilakukan dengan pemrograman bahasa python. Selanjutnya dilakukan tahap pengujian terhadap sistem yang diimplementasi untuk mengetahui keberhasilan sistem menggunakan metode-metode dalam penelitian.

3.1 Menjalankan *Cloud Programming*

Pada penelitian ini menggunakan *Cloud programming* dengan Google Colab, bagian ini berfungsi untuk *import* Google Colab kedalam *drive, library* dan melihat hasil pengambilan notasi kedalam Google Colab. Dalam penggunaan pengambilan notasi menggunakan link dari *roboflow* untuk melakukan pemindahan maupun penyimpanan data notasi yang sudah melalui tahapan pengenalan dengan *labelimg* setelah itu hasil link dari *roboflow* akan ditetapkan kedalam Google Colab.

Tabel 1. Pengambilan Hasil Pengenalan Notasi Alpha

Flowchart	
Source Code	2. from google.colab import drive drive.mount ('/content/drive') 3. !git clone https://github.com/roboflow-ai/darknet.git 4. !wget https://github.com/AlexeyAB/darknet/releases/download/ darknet_yolo_v4_pre/yolov4-tiny.weights !wget https://github.com/AlexeyAB/darknet/releases/download/ darknet_yolo_v4_pre/yolov4-tiny.conv.29 5. !curl - L "https://app.roboflow.com/ds/GUZLRxAcWL?key=WkTzUeg5Bd" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip

Tabel 1. Pengambilan Hasil Pengenalan Notasi Alpha (Lanjutan)

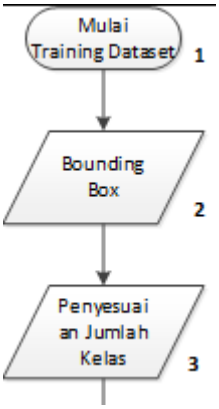
Hasil keluaran	`/content/darknet`
	`  % Total  % Received % Xferd  Average Speed  Time  Time  Time  Current`
	`  Dload  Upload  Total  Spent  Left  Speed`
	`100  890  100  890  0  0  1068  0 --:--:-- --:--:-- --:--:--  1067`
	`100 10.6M  100 10.6M  0  0  3287k  0  0:00:03  0:00:03 --:--:-- 7853k`
	`Archive:  roboflow.zip`
	` extracting: README.roboflow.txt`
	`  creating: test/`
	` extracting: test/0001-21-_jpg.rf.f41fe6ec9fe7c7ad0dd75b9c28ae9fed.jpg`
	` extracting: test/0001-21-_jpg.rf.f41fe6ec9fe7c7ad0dd75b9c28ae9fed.txt`
	` extracting: test/0001-7--n1_jpg.rf.d59082b0ced817a15c06f3c6235c4c15.jpg`
	` extracting: test/0001-7--n1_jpg.rf.d59082b0ced817a15c06f3c6235c4c15.txt`
	` extracting: test/0001-8-_jpg.rf.f89ac5b4491ccfb04bb86425cd4c341b.jpg`
	` extracting: test/0001-8-_jpg.rf.f89ac5b4491ccfb04bb86425cd4c341b.txt`
	` extracting: test/0001-9-_jpg.rf.0a839a31a8fe1ec0a859ebba53380a53.jpg`
	` extracting: test/0001-9-_jpg.rf.0a839a31a8fe1ec0a859ebba53380a53.txt`
	` extracting: test/4_ketuk_n1_png_jpg.rf.6a57c978f393a36b2f42d8fb57b0a5a0.jpg`
	` extracting: test/4_ketuk_n1_png_jpg.rf.6a57c978f393a36b2f42d8fb57b0a5a0.txt`
	` extracting: test/_darknet.labels`
	`  creating: train/`
	` extracting: train/0001-1-_jpg.rf.066b1a8959090b4818d7b40fd906fc3e.jpg`
	` extracting: train/0001-1-_jpg.rf.066b1a8959090b4818d7b40fd906fc3e.txt`

Pada Tabel 1 merupakan hasil dari pengambilan notasi yang mengacu pada *block diagram* dan *data flow digram* citra latih.

3.2 Penyesuaian Jumlah Kelas

Pada pengujian untuk penyesuaian jumlah kelas ini dilakukan pemasukan jumlah kelas sebanyak 16 kelas yang terdiri dari notasi treble clef atau tangan kanan yang terdiri dari oktaf 3 hingga 5 dan notasi bass clef atau tangan kiri terdiri dari oktaf 1 hingga 2 dengan total sebanyak 563 data gambar.

Tabel 2. Penyesuaian Jumlah Kelas

Flowchart	

Tabel 2. Penyesuaian Jumlah Kelas (Lanjutan)

Source Code 1	%cd /content/darknet/
	%cp train/_darknet.labels data/obj.names
	%mkdir data/obj
	
	%cp train/*.jpg data/obj/
	%cp valid/*.jpg data/obj/
	
	%cp train/*.txt data/obj/
	%cp valid/*.txt data/obj/
	
	with open('data/obj.data', 'w') as out:out.write('classes = 16\n') out.write('train = data/train.txt\n')out.write('valid = data/valid.txt\n')out.write('names = data/obj.names\n') out.write('backup = backup/')
	
	import os
	
	with open('data/train.txt', 'w') as out:
	for img in [f for f in os.listdir('train') if f.endswith('jpg')]:
	out.write('data/obj/' + img + '\n')import os
	
	with open('data/valid.txt', 'w') as out:
	for img in [f for f in os.listdir('valid') if f.endswith('jpg')]:
	out.write('data/obj/' + img + '\n')

Source Code 1	def file_len(fname): with open(fname) as f: for i, l in enumerate(f):pass return i + 1
Source Code 2	num_classes = file_len('train/_darknet.labels') max_batches = num_classes*2000 steps1 = .8 * max_batchessteps2 = .9 * max_batches steps_str = str(steps1)+','+str(steps2)num_filters = (num_classes + 5) * 3 print("writing config for a custom YOLOv4 detector detecting number of classes: " + str(num_classes)) if os.path.exists('./cfg/custom-yolov4-tiny-detector.cfg'): os.remove('./cfg/custom-yolov4-tiny-detector.cfg') f writetemplate(line, cell): with open(line, 'w') as f: f.write(cell.format(**globals()))

Tabel 2. Penyesuaian Jumlah Kelas (Lanjutan)

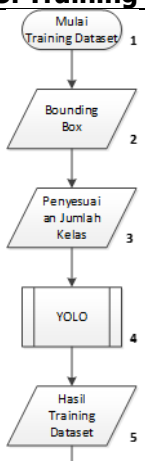
Hasil keluaran	`writing config for a custom YOLOv4 detector detecting number of classes: 16`

Pada Tabel 2 source code 1 merupakan source code untuk mengalokasi dataset kedalam *google drive* dan menentukan banyaknya jumlah kelas yang dimasukan kedalam *souce code* YOLO. Dalam *source* code tersebut terdapat objek gambar yang dibatasi dengan format *.jpg* untuk menyelaraskan format yang dimasukan, dan objek gambar yang berformat *.jpg* disimpan pada data darknet untuk diproses. Langkah awal dalam mengalokasi data yaitu menyalin data obj.names kedalam file bernama obj. Selanjutnya menyalin format gambar *.jpg* dari file train dan valid dan meyalin data berupa format *.txt*. Jika data sudah disalin maka tahap berikutnya membuat data *.txt* didalam file bernama *obj* untuk menentukan alokasi data seperti jumlah kelas, penyimpanan *data train.txt*, data *valid.txt*, dan file *backup*. Pada tabel 3. 2 *source code* 2 merupakan perhitungan untuk jumlah kelas yang telah di masukan seperti pada *source code* 1 sebanyak 16 kelas, setelah itu jumlah kelas di proses untuk dimasukan kedalam *source code* YOLO. Langkah awal dalam perhitungan jumlah kelas pada *source code* 2 yaitu dengan cara jumlah kelas sebelumnya pada *source code* 1 dimasukan kedalam objek bernama *max_batches, steps_str,* dan *num_filters*. Setelah melakukan perhitungan maka data *custom-yolov4-tiny-detector.cfg* dimasukan kedalam data yang telah dilakukan kedalam data tersebut. Perhitungan ini sebagai berikut, didalam *darknet.labels* tersebut berisi nama dari hasil label notasi yang berjumlah 16 kelas, selanjutnya dari 16 kelas tersebut dilakukan pehitungan max_batches (16*2000 = 32000) yang berfungsi untuk waktu total dalam melakukan training, selanjutya melakukan perhitungan untuk steps1 (0.8*32000 = 25.600) dan steps2 (0,9*32000 = 28.800) yang berfungsi sebagai perulangan kecepatan pembelajaran dalam YOLO ditentukan oleh faktor pengali dari scale, selanjutnya num_filters((16+5)*3 = 63).

3.3 *Training Model*

Pada pengujian training model ini melakukan pengujian terhadap sistem untuk melakukan training dalam Google Colab. Training model ini bertujuan untuk notasi balok agar dapat menghasilkan hasil konversi.

Tabel 3. Training model Alpha

Flowchart	

Tabel 3. Training model Alpha (Lanjutan)

Source Code	`!./darknet detector train data/obj.data cfg/custom-yolov4-tiny-detector.cfg yolov4-tiny.conv.29 -dont_show -map`

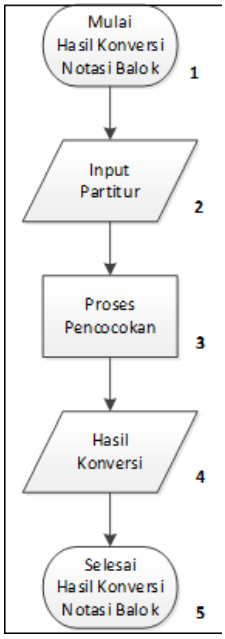
Hasil keluaran	`[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05`
	`nms_kind: greedynms (1), beta = 0.600000`
	`31 route 27 -> 13 x 13 x 256`
	`32 conv 128 1 x 1/ 1 13 x 13 x 256 -> 13 x 13 x 128 0.011 BF`
	`33 upsample 2x 13 x 13 x 128 -> 26 x 26 x 128`
	`34 route 33 23 -> 26 x 26 x 384`
	`35 conv 256 3 x 3/ 1 26 x 26 x 384 -> 26 x 26 x 256 1.196 BF`
	`36 conv 63 1 x 1/ 1 26 x 26 x 256 -> 26 x 26 x 63 0.022 BF`
	`37 yolo`
	`[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05`
	`nms_kind: greedynms (1), beta = 0.600000`
	`Total BFLOPS 6.811`
	`avg_outputs = 301665`
	`Allocate additional workspace_size = 26.22 MB`
	`Loading weights from yolov4-tiny.conv.29...`
	`seen 64, trained: 0 K-images (0 Kilo-batches_64)`
	`Done! Loaded 29 layers from weights-file`
	`Learning Rate: 0.00261, Momentum: 0.9, Decay: 0.0005`
	`Create 6 permanent cpu-threads`
	`Loaded: 0.688628 seconds`

Pada Tabel 3 merupakan hasil dari pengujian tiap notasi dengan menggunakan Google Colab dengan acuan pada *flowchart training model*.

3.4 HASIL KONVERSI

Pada tahap ini merupakan hasil konversi dari pengujian dengan menggunakan dataset untuk testing lagu notasi balok.

Tabel 4. Hasil Konversi

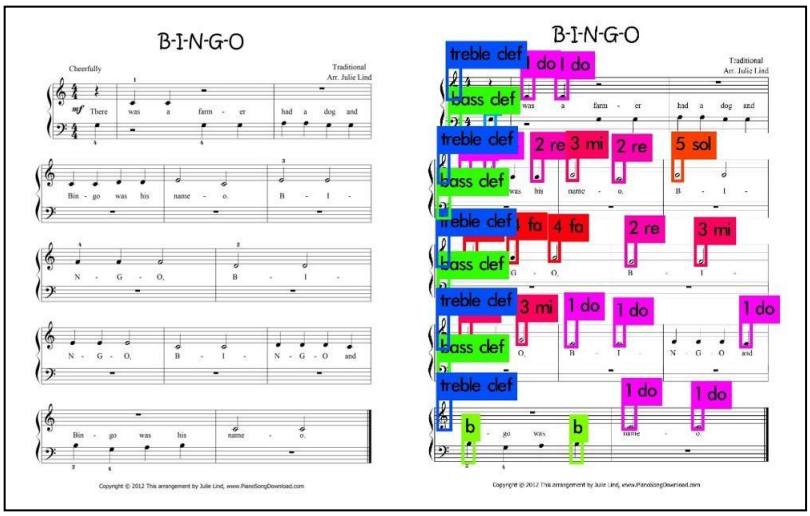
Flowchart	

Tabel 4. Hasil Konversi (Lanjutan)

Source Code	`#/test image`
	`test_images = [f for f in os.listdir('test') if f.endswith('Bingo.jpg')]`
	`import random`
	`img_path = "test/" + random.choice(test_images);`
	
	`#test out`
	`!./darknet detect cfg/custom-yolov4-tiny-detector.cfg backup/custom-yolov4-tiny-detector_best.weights {img_path} -dont-show`
	`imShow('predictions.jpg')`

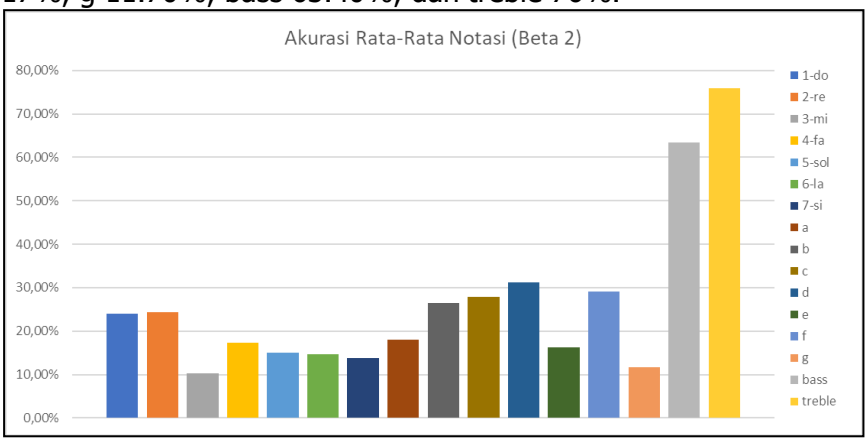
Hasil keluaran	

Pada Tabel 4. merupakan proses pengambilan gambar yang dideteksi dan hasil dari pendeteksian lagu notasi balok. Dapat disimpulkan pada hasil dari pengujian dengan lagu yang berjudul B-I-N-G-O bahwa hasil pendeteksian dapat melakukan mendeteksi lagu dengan notasi balok dan mengkonversi menjadi notasi angka, pada Gambar 5 merupakan konversi notasi balok berupa partirtur lagu menjadi notasi angka.



Gambar 5. Hasil Pengujian Beta 2

Pada Gambar 5 merupakan hasil dari pengujian yang dilakukan sebanyak 8 kali pengujian. Dalam pengujian beta digunakan dengan cloud programming dengan Google Colab sebagai platform dengan dataset per-notasi sebanyak 16 buah kelas dan jumlah total gambar notasi balok atau dataset sebanyak 563. Pengujian ini membuahkan hasil rata-rata akurasi tiap notasi sebesar 1-do sebesar 24.07%, 2-re 24.38%, 3-mi 10.34%, 4-fa 17.38%, 5-sol 15.09%, 6-la 14.62%, 7-si 13.77%, a 17.93%, b 26.39%, c 27.92%, d 31.25%, e 16.17%, f 29.17%, g 11.70%, bass 63.40%, dan treble 76%.



Gambar 5. Grafik Pengujian Notasi Lagu

4. KESIMPULAN

Berdasarkan hasil pengujian diatas yang telah dilakukan dapat dibuktikan bahwa algoritma YOLO dapat digunakan pada konversi notasi balok menjadi notasi angka hal ini dapat dinyatakan bahwa dalam pengujian beta 1 yang dilakukan, algoritma YOLO dapat diterapkan dalam koversi notasi balok menjadi notasi angka, adapun hasil tingkat akurasi untuk melodi tertinggi adalah notasi 1 do sebesar 85.07%, dan terendah 7 si 66.49%, sedangkan untuk chord tertinggi adalah notasi c 91.18%, dan terendah b 40% dalam hasil akurasi ini pada notasi do dan chord c memiliki kedudukan atau ciri yang berbeda di garis paranada sedangkan pada notasi 7 si dan chord b memiliki kedudukan dan ciri yang sama dalam garis paranada. Berdasarkan pengujian beta 2 notasi yang sering muncul dalam partirut lagu Bingo notasi angka diperoleh pada notasi 1 do dengan akurasi 93.67% dan notasi yang sering muncul pada bagian chord akurasi diperoleh chord b sebesar 63.33%. Hasil dari akurasi yang diperoleh ini berasalan dari training yang dilakukan sebanyak 15 kali untuk beta 1 dan 8 kali untuk beta 2, dari hasil tersebut menghasilkan gambar yang terdeteksi berupa konversi notasi angka dan pembedahan *chord*, namun dalam pendeteksian objek beberapa notasi masih terdeteksi menjadi satu objek pendeteksian yang sama dan tidak menutup kemungkinan kesalahan terjadi pada saat konversi notasi balok menjadi notasi angka, hal ini disebabkan oleh adanya kedudukan notasi balok antara melodi dan *chord* yang sama.

DAFTAR PUSTAKA

Chrisantyo A.A, L., Wijana, K., & Restyandito. (2007). Program Konversi Not Balok Dengan Struktur Musicxml Ke Not Angka. Seminar Nasional Dosen Stmik Amikom Yogyakarta, 2007(2007: *Seminar Nasional Teknologi*), 1–11. http://journal.amikom.ac.id/index.php/SN/article/view/2112

Chrisantyo, L., Hartanto, R., & Nugroho, L. E. (2012). Pengembangan Aplikasi Konversi Representasi Not Balok Ke Not Angka Untuk Paduan Suara Campur. *Jurnal*

Informatika,8(1). https://doi.org/10.21460/inf.2012.81.115

Hajic, J., & Pecina, P. (2017). The MUSCIMA++ Dataset for Handwritten Optical Music Recognition. *Proceedings of the International Conference on Document Analysis and Recognition*, ICDAR, 1(Vi), 39–46. https://doi.org/10.1109/ICDAR.2017.16

Higgins, L. (2012). *Community Music: In Theory and In Practice*. In Community Music: In Theory and In Practice. https://doi.org/10.1093/acprof:oso/9780199777839.001.0001

I Wayan Aldon Alba Dona. (2021). *Konversi Not Balok Menjadi Not Angka Dengan Metode OCR. Konversi Not Balok Menjadi Not Angka Dengan Metode OCR*, 75. https://arxiv.org/pdf/1707.06526.pdf%0Ahttps://www.yrpri.org%0Ahttp://weekly.cnbnews.com/news/article.html?no=124000%0Ahttps://www.fordfoundation.org/%0Ahttp://bibliotecavirtual.clacso.org.ar/Republica_Dominicana/ccp/20120731051903/prep%0Ah t tp://webpc.cia

Manus, M., Palmer, W. A., & Lethco, A. V. (2010*). Lesson Book Level 1*. Alfred Basic Piano Library, 7.

Reisenweaver, A. (2012). Guido of Arezzo and His Influence on Music Learning. *Musical Offerings*, 3(1), 37–59. https://doi.org/10.15385/jmo.2012.3.1.4