

**DEEP LEARNING OBJECT DETECTION PADA VIDEO  
MENGUNAKAN TENSORFLOW DAN CONVOLUTIONAL NEURAL  
NETWORK**

(Studi Kasus: Klasifikasi Gambar Meja dan Kursi Motif Ukiran Jepara)

**TUGAS AKHIR**

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana  
Program Studi Statistika



**SYARIFAH ROSITA DEWI**

**14 611 242**

**PROGRAM STUDI STATISTIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS ISLAM INDONESIA  
YOGYAKARTA**

**2018**

**HALAMAN PERSETUJUAN PEMBIMBING**

**TUGAS AKHIR**

Judul : *Deep Learning Object Detection* Pada Video  
Menggunakan *Tensorflow* dan *Convolutional  
Neural Network*

Nama Mahasiswa : Syarifah Rosita Dewi

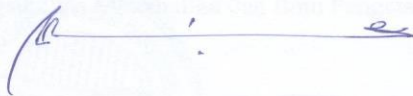
Nomor Mahasiswa : 14 611 242

**TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK  
DIUJIKAN**

Yogyakarta, 17 April 2018

الجامعة الإسلامية  
الاستاذة الإندونيسية

**Pembimbing**



**(Dr. RB. Fajriya Hakim, S.Si., M.Si.)**

**HALAMAN PENGESAHAN**

**TUGAS AKHIR**

**DEEP LEARNING OBJECT DETECTION PADA VIDEO  
MENGUNAKAN TENSORFLOW DAN CONVOLUTIONAL NEURAL  
NETWORK**

**Nama Mahasiswa: Syarifah Rosita Dewi**

**Nomor Mahasiswa: 14 611 242**

**TUGAS AKHIR INI TELAH DIUJIKAN  
PADA TANGGAL 25 MEI 2018**

**Nama Penguji**

1. Ir. Ali Parkhan M.T.
2. Tuti Purwaningsih, S.Stat., M.Si.
3. Dr. RB. Fajriya Hakim, S.Si., M.Si.

**Tanda tangan**

(.....)  
(.....)  
(.....)

الجامعة الإسلامية  
Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam



(Drs. Aliwar, M.Sc., Ph.D.)

## KATA PENGANTAR



*Assalamu'alaikum Wr. Wb.*

*Alhamdulillah* rabbil'alamiin, puji syukur senantiasa penulis panjatkan atas kehadiran Allah SWT yang telah melimpahkan rahmat dan hidayahNya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul “*Deep Learning Object Detection* Pada Video Menggunakan *Tensorflow* dan *Convolutional Neural Network*” sebagai salah satu persyaratan yang harus dipenuhi dalam menyelesaikan jenjang Strata Satu atau S1 di Jurusan Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia. Shalawat serta salam *Insyallah* selalu tercurah kepada Nabi Muhammad SAW serta para sahabat dan pengikutnya sampai akhir zaman.

Penyelesaian tugas akhir ini tidak terlepas dari bantuan, arahan, dan bimbingan dari berbagai pihak. Untuk itu penulis mengucapkan terima kasih kepada:

1. Bapak Nandang Sutrisno, SH., LL.M., M.Hum., Ph.D., selaku rektor Universitas Islam Indonesia.
2. Bapak Drs. Allwar, Ph. D selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia, Yogyakarta.
3. Bapak Dr. RB. Fajriya Hakim, S.Si, M.Si. selaku Ketua Jurusan Statistika sekaligus dosen pembimbing Tugas Akhir, yang telah banyak meluangkan waktu, memberikan masukan yang membangun serta arahan yang positif saat melakukan bimbingan dan selalu memberikan ilmunya.
4. Seluruh staf pengajar Program Studi Statistika Universitas Islam Indonesia yang telah memberikan bekal ilmu kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini.
5. Orang tua penulis, Bapak Waijo dan Ibu Rusmiyati yang selalu luar biasa mendoakan dan pantang menyerah bekerja keras demi kelancaran studi penulis.

6. Kakak-Adik penulis, yaitu Anisa Rosmala Dewi dan Rosiana Rahma Dewi, serta keluarga lainnya yang selalu mendoakan dan memberikan semangat kepada penulis.
7. Teman-teman sepermainan dan seperjuangan yaitu Ridha, Bila, Leni, Aul, Panji, Ditia, Erdwika yang selalu membuat rusuh tetapi saling mengingatkan untuk menyelesaikan studi dalam menyelesaikan tugas akhir ini serta memberikan semangat satu sama lain.
8. Dewi dan Cindy yang menemani dan menghabiskan waktu bersama untuk menyelesaikan tugas akhir ini selama di perpustakaan UII.
9. Seluruh teman-teman satu bimbingan tugas akhir, yang senantiasa berbagi ilmu dan informasi yang bermanfaat sampai tugas akhir ini selesai.
10. Teman-teman Statistika UII Angkatan 2014 yang sedang berjuang bersama untuk meraih gelar S.Stat dan Toga UII, terimakasih atas pengalaman berharga selama menjadi mahasiswa Statistika UII.
11. Semua pihak yang tidak dapat penulis sebutkan satu per satu, terima kasih.

Semoga segala bantuan, bimbingan, dan pengajaran yang telah diberikan kepada penulis mendapatkan imbalan dari Allah SWT. Penulis mohon maaf apabila selama proses penyusunan tugas akhir ini terdapat kekhilafan dan kesalahan. Penulis menyadari sepenuhnya keterbatasan kemampuan dalam penulisan tugas akhir ini, oleh karena itu penulis mengharap adanya kritik dan saran yang membangun demi kesempurnaan penyusunan dan penulisan tugas akhir ini. Akhir kata, semoga tugas akhir ini dapat bermanfaat bagi semua yang membaca dan membutuhkan, Aamiin aamiin ya robbal'alamiin.

*Wassalamu'alaikum, Wr. Wb.*

Yogyakarta, Maret 2018

Penulis

## DAFTAR ISI

<b>HALAMAN JUDUL</b> .....	i
<b>HALAMAN PERSETUJUAN PEMBIMBING</b> .....	ii
<b>HALAMAN PENGESAHAN</b> .....	iii
<b>KATA PENGANTAR</b> .....	iv
<b>DAFTAR ISI</b> .....	vi
<b>DAFTAR TABEL</b> .....	x
<b>DAFTAR GAMBAR</b> .....	xi
<b>DAFTAR LAMPIRAN</b> .....	xiii
<b>PERNYATAAN</b> .....	xiv
<b>INTISARI</b> .....	xv
<b>ABSTRACT</b> .....	xvi
<b>BAB I PENDAHULUAN</b> .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Batasan Masalah .....	4
1.4 Tujuan Penelitian .....	4
1.5 Manfaat Penelitian .....	5
<b>BAB II TINJAUAN PUSTAKA</b> .....	6
<b>BAB III LANDASAN TEORI</b> .....	11
3.1 Meja dan Kursi .....	11
3.2 Ukiran Jepara .....	11
3.2.1 Sejarah .....	11
3.2.2 Motif Ukiran .....	12
3.3 Pengolahan Citra .....	13
3.3.1 Definisi Citra .....	13
3.3.2 Definisi Citra Digital .....	14
3.3.3 Tipe Citra Digital .....	14
3.3.4 Ekstraksi Ciri Suatu Gambar .....	15
3.4 <i>Web Crawler</i> .....	16

3.5 <i>Artificial Intelligence (Kecerdasan Buatan)</i> .....	17
3.6 <i>Computer Vision</i> .....	17
3.7 <i>Machine Learning</i> .....	17
3.8 <i>Deep Learning</i> .....	18
3.9 <i>Image Processing (Pengolahan Citra)</i> .....	19
3.10 <i>Segmentasi</i> .....	19
3.11 <i>Pattern Recognition</i> .....	19
3.12 <i>Object Detection</i> .....	20
3.13 <i>Artificial Neural Network</i> .....	20
3.13.1 <i>Multilayer Networks</i> .....	22
3.13.2 <i>Backpropagation</i> .....	23
3.13.3 <i>Tipe Fungsi Aktivasi</i> .....	23
3.14 <i>Learning Rate</i> .....	24
3.15 <i>ReLU (Rectrified Liniear Unit)</i> .....	24
3.16 <i>Dropout Regularization</i> .....	25
3.17 <i>Convolutional Neural Network</i> .....	26
3.17.1 <i>Convolution Layer (Conv. Layer)</i> .....	26
3.17.2 <i>Stride</i> .....	27
3.17.3 <i>Padding</i> .....	27
3.17.4 <i>Crossentropy Loss Function</i> .....	27
3.17.5 <i>Pooling Layer</i> .....	28
3.17.6 <i>Activation Function</i> .....	28
3.17.7 <i>Arsitektur Jaringan CNN</i> .....	28
3.18 <i>Python</i> .....	30
3.19 <i>Tensorflow</i> .....	30
<b>BAB IV METODOLOGI PENELITIAN</b> .....	32
4.1 <i>Populasi dan Sampel Penelitian</i> .....	32
4.2 <i>Variabel dan Definisi Operasional Penelitian</i> .....	32
4.3 <i>Jenis dan Sumber Data</i> .....	32
4.4 <i>Metode Analisa Data</i> .....	32
4.5 <i>Tahapan Penelitian</i> .....	33

<b>BAB V</b>	<b>ANALISIS DAN PEMBAHASAN</b>	34
5.1	Pengumpulan <i>Dataset</i>	34
5.1.1	Program <i>Javascript</i>	34
5.1.2	Program <i>Python</i>	36
5.2	<i>Preprocessing</i> Citra	38
5.2.1	Pelabelan Gambar/Citra	39
5.2.2	Konversi <i>Dataset</i> Meta XML ke CSV	39
5.2.3	Konversi <i>Dataset</i> CSV ke <i>TFRecord</i>	40
5.2.4	<i>Label Map</i>	40
5.3	Pengolahan Citra	41
5.3.1	Konfigurasi <i>Object Detection Training Pipeline</i>	41
5.3.2	<i>Training Neural Network</i>	42
5.3.3	<i>Export Graph Model</i>	42
5.3.4	Arsitektur Jaringan CNN	43
5.3.4.1	<i>Convolution Layer</i>	43
5.3.4.2	<i>Activation Function</i>	45
5.3.4.3	<i>Pooling Layer</i>	45
5.3.4.4	<i>Fully Connected Layer</i>	47
5.3.4.5	<i>Classification</i>	48
5.3.4.6	<i>Detection Output</i>	48
5.4	Model Hasil <i>Training</i>	48
5.4.1	<i>Training Steps</i>	48
5.4.2	Total <i>Loss</i>	49
5.4.3	<i>Tensor Graph</i>	49
5.4.3.1	<i>Batch</i>	50
5.4.3.2	<i>Train Step</i>	50
5.4.3.3	Total <i>Loss</i>	51
5.4.3.4	<i>Global Step</i>	51
5.4.4	Model	52
5.5	Hasil Deteksi	52
<b>BAB VI</b>	<b>KESIMPULAN DAN SARAN</b>	55



6.1 Kesimpulan.....	55
6.2 Saran.....	55
DAFTAR PUSTAKA .....	56
LAMPIRAN.....	60

## DAFTAR TABEL

<b>Tabel 2.1</b>	Perbandingan dengan Penelitian Terdahulu .....	9
<b>Tabel 4.1</b>	Definisi operasional variabel .....	32

## DAFTAR GAMBAR

<b>Gambar 3.1</b> Contoh Ukiran pada Kursi .....	13
<b>Gambar 3.2</b> Contoh Ukiran pada Meja .....	13
<b>Gambar 3.3</b> Ilustrasi Neuron dengan Model Matematisnya.....	20
<b>Gambar 3.4</b> <i>Artificial Neuron</i> .....	21
<b>Gambar 3.5</b> <i>Multilayer Neural Network</i> .....	22
<b>Gambar 3.6</b> <i>Backpropagation</i> .....	23
<b>Gambar 3.7</b> Grafik Fungsi Aktifasi ReLu .....	25
<b>Gambar 3.8</b> <i>Before and After Dropout</i> .....	25
<b>Gambar 3.9</b> Proses <i>Max Pooling</i> .....	28
<b>Gambar 3.10</b> <i>Image RGB</i> .....	29
<b>Gambar 3.11</b> <i>Feature Map</i> .....	29
<b>Gambar 4.1</b> <i>Diagram Alir Penelitian</i> .....	33
<b>Gambar 5.1</b> <i>Google Image Search</i> .....	34
<b>Gambar 5.2</b> <i>Developer Mode Console</i> .....	35
<b>Gambar 5.3</b> <i>Create Variable</i> .....	35
<b>Gambar 5.4</b> <i>Download URL</i> .....	36
<b>Gambar 5.5</b> <i>Import Packages dan Membuat Argument</i> .....	36
<b>Gambar 5.6</b> <i>Perulangan Download URL</i> .....	37
<b>Gambar 5.7</b> <i>Perintah pada Python</i> .....	38
<b>Gambar 5.8</b> <i>Dataset Citra Meja Motif Ukiran</i> .....	38
<b>Gambar 5.9</b> <i>Dataset Citra Kursi Motif Ukiran</i> .....	38
<b>Gambar 5.10</b> <i>Proses Pelabelan Gambar Meja</i> .....	39
<b>Gambar 5.11</b> <i>Proses Pelabelan Gambar Kursi</i> .....	39
<b>Gambar 5.12</b> <i>Kode Konversi XML ke CSV</i> .....	40
<b>Gambar 5.13</b> <i>Kode Konversi CSV ke TFRecord</i> .....	40
<b>Gambar 5.14</b> <i>Kode Konfigurasi Label Map</i> .....	41
<b>Gambar 5.15</b> <i>Kode Konfigurasi Pipeline</i> .....	41
<b>Gambar 5.16</b> <i>Kode Program Proses Training</i> .....	42
<b>Gambar 5.17</b> <i>Kode Program Export Graph Model</i> .....	42

<b>Gambar 5.18</b> Arsitektur Jaringan .....	43
<b>Gambar 5.19</b> <i>Convolutional Layer</i> .....	43
<b>Gambar 5.20</b> Proses Konvolusi .....	45
<b>Gambar 5.21</b> Posisi Proses Konvolusi.....	45
<b>Gambar 5.22</b> <i>Pooling Layer</i> .....	45
<b>Gambar 5.23</b> <i>Fully Connected Layer</i> .....	46
<b>Gambar 5.24</b> Grafik <i>Global Training Step</i> .....	47
<b>Gambar 5.25</b> Grafik <i>Total Loss</i> .....	48
<b>Gambar 5.26</b> <i>Graph Legend</i> .....	48
<b>Gambar 5.27</b> <i>Batch Graph</i> .....	49
<b>Gambar 5.28</b> <i>Train Step Graph</i> .....	49
<b>Gambar 5.29</b> <i>Total Loss Graph</i> .....	50
<b>Gambar 5.30</b> <i>Global Step Graph</i> .....	50
<b>Gambar 5.31</b> Direktori Model Hasil <i>Training</i> .....	51
<b>Gambar 5.32</b> Hasil Deteksi Kursi Motif Ukiran Jepara .....	53
<b>Gambar 5.33</b> Hasil Deteksi Meja Motif Ukiran Jepara .....	53
<b>Gambar 5.34</b> Hasil Deteksi Meja dan Kursi Motif Ukiran Jepara .....	54
<b>Gambar 5.35</b> Hasil Deteksi Menggunakan Video .....	54

## DAFTAR LAMPIRAN

<b>Lampiran 1</b> <i>Script Javascript</i> .....	60
<b>Lampiran 2</b> <i>Script Python Crawling Data</i> .....	61
<b>Lampiran 3</b> <i>Script Konversi Datasets Meta XML ke CSV</i> .....	63
<b>Lampiran 4</b> <i>Script Konversi Datasets CSV ke TFRecord</i> .....	64
<b>Lampiran 5</b> <i>Script Label Map</i> .....	66
<b>Lampiran 6</b> <i>Script Konfigurasi Object Detection Pelatihan Pipeline</i> .....	67
<b>Lampiran 7</b> <i>Training Neural Network</i> .....	71
<b>Lampiran 8</b> <i>Script Export Graph Model</i> .....	74
<b>Lampiran 9</b> <i>Script Deteksi Gambar</i> .....	75
<b>Lampiran 10</b> <i>Script Deteksi Video</i> .....	78

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang diaacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, April 2018



Penulis

# DEEP LEARNING OBJECT DETECTION PADA VIDEO MENGUNAKAN TENSORFLOW DAN CONVOLUTIONAL NEURAL NETWORK

(Studi Kasus: Klasifikasi Gambar Meja dan Kursi Motif Ukiran Jepara)

Syarifah Rosita Dewi

Program Studi Statistika Fakultas MIPA

Universitas Islam Indonesia

## INTISARI

Indonesia merupakan salah satu negara yang terkenal memiliki berbagai macam kebudayaan khas. Hampir setiap daerah memiliki kebudayaan khas yang unik salah satunya adalah kota Jepara yang terkenal dengan sebutan “kota ukir”. Ukiran Jepara memiliki ciri khas yaitu adanya motif jumbai dan banyak digunakan untuk memperindah *furniture* seperti meja dan kursi. Meja dan kursi merupakan dua perabot rumah tinggal yang sangat populer. Keduanya hampir selalu ada dan berfungsi vital dalam setiap aktivitas di dalam ruangan maupun di luar ruangan. Dengan berkembangnya zaman yang semakin modern, desain meja dan kursi juga semakin bervariasi yang membuat produsen meja dan kursi saling berlomba menghadirkan desain terbaik dengan bentuk yang semakin modern juga. Agar ukiran Jepara tidak dilupakan maka diperlukan sistem yang dapat mengenali meja dan kursi motif ukiran Jepara supaya tidak menghilang karena adanya jenis meja dan kursi dengan penerapan tren gaya terkini (*modern*). Salah satu bidang penelitian yang masih berkembang sampai saat ini adalah kecerdasan buatan (*Artificial Intelligence*). Pengembangan cabang ilmu *AI*, salah satunya adalah *computer vision*. Dalam *computer vision* terdapat permasalahan yaitu *object detection* dan *image classification*. *Deep learning* yang digunakan untuk pengenalan dan klasifikasi objek adalah *Convolutional Neural Network* karena banyak digunakan pada penelitian terdahulu dan menghasilkan hasil yang signifikan dalam pengenalan citra. Pada penelitian ini dilakukan pengenalan objek meja dan kursi motif ukiran Jepara menggunakan *framework Tensorflow* dengan *dataset* sebanyak 500 gambar. Hasil penelitian menunjukkan bahwa dengan metode *CNN* didapatkan tingkat akurasi hingga 98% untuk melakukan deteksi meja dan kursi motif ukiran Jepara pada sebuah *frame* gambar dan video.

**Kata kunci:** *Artificial Intelligence, Computer Vision, Deep Learning, Convolution Neural Network, Furniture Ukiran Jepara, Object Detection, Tensorflow*

# **DEEP LEARNING OBJECT DETECTION ON VIDEO USING TENSORFLOW AND CONVOLUTIONAL NEURAL NETWORK**

(Case Study: Classification Images Table and Chair of Carving Motif Jepara)

Syarifah Rosita Dewi

Department of Statistics, Faculty of Mathematics and Natural Science

Islamic University of Indonesia

## **ABSTRACT**

Indonesia is one of the famous countries that has a variety of distinctive culture. Almost every region has a unique culture which is the city of Jepara known as "carving town". Jepara carving has a characteristic that is a tassel motif and widely used to embellish furniture such as tables and chairs. Tables and chairs are two very popular home furnishings. Both are almost always present and function vital in any activity indoors or outdoors. With the development of an increasingly modern era, the design of desks and chairs are also increasingly varied which makes table and chair manufacturers competing each other to bring the best design with a more modern form as well. So that Jepara carving is not forgotten then it is necessary system that can recognize table and chair Jepara carving motif so that not disappearing because existence of type of table and chair with applying trend modern (modern) style. One area of research that is still up to date is artificial intelligence. Development of science branch of AI, one of them is computer vision. In computer vision there are problems of object detection and image classification. Deep learning used for object recognition and classification is Convolutional Neural Network method because it is widely used in previous research and produces significant results in image recognition. In this research, the introduction of table and chair object of Jepara carving motif using Tensorflow framework with dataset of 500 images. The results showed that the CNN method obtained an accuracy of up to 98% for the detection of Jepara carving table and chair motifs on an image frame and video.

**Keyword:** *Artificial Intelligence, Computer Vision, Deep Learning, Convolution Neural Network, Furniture Carving Jepara, Object Detection, Tensorflow*



# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Indonesia merupakan salah satu negara yang terkenal kaya akan kesenian, kebudayaan dan kerajinan tradisionalnya. Bukan hanya satu daerah saja yang memiliki kebudayaan khas, namun hampir setiap daerah memiliki ciri khas seni dan budaya yang berbeda-beda. Salah satu daerah yang memiliki kebudayaan khas adalah kota Jepara yang terletak di kawasan pantai utara Jawa Tengah. Kota Jepara terkenal dengan sebutan “kota ukir” yang menjadi salah satu aset berharga dari seni dan budaya yang ada di Indonesia.

Setiap daerah mempunyai ciri khas ukiran masing-masing begitu juga dengan ukiran Jepara yang mempunyai motif dan corak yang unik. Ciri khas ukiran Jepara adalah adanya motif jumbai. Jumbai adalah daun yang terbuka seperti kipas dan setiap pangkal daun jumbai biasanya terdapat buah wuni. Ukiran di Jepara pada umumnya banyak digunakan untuk memperindah sebuah mebel/*furniture* seperti pada meja, kursi dan lain sebagainya.

Meja dan kursi merupakan dua perabot rumah tinggal yang sangat populer. Keduanya hampir selalu ada dan berfungsi vital dalam setiap aktivitas di dalam ruangan maupun di luar ruangan. Dengan berkembangnya zaman yang semakin modern, desain meja dan kursi juga semakin berkembang dengan adanya perubahan bentuk yang semakin bervariasi. Beberapa produsen meja dan kursi mulai menjamur dan saling berlomba menghadirkan desain terbaik untuk dipersembahkan kepada para pemilik rumah begitupun dengan produsen penghasil meja dan kursi motif ukiran Jepara.

Masyarakat umum maupun konsumen terkadang memilih meja dan kursi selain dari kualitas juga berdasarkan keunikan dari meja dan kursi tersebut. Permasalahan yang terjadi sekarang adalah generasi muda, konsumen atau masyarakat umum kurang mengetahui motif ukiran yang

terdapat pada meja dan kursi yang dibelinya. Oleh karena itu, diperlukan sistem yang dapat mengenali meja dan kursi motif ukiran Jepara. Ukiran merupakan warisan budaya Indonesia yang seharusnya tetap dilestarikan supaya tetap dikenal oleh masyarakat di masa mendatang agar motif ukiran pada meja dan kursi tidak menghilang dan dilupakan karena adanya jenis meja dan kursi dengan penerapan tren gaya terkini (*modern*).

Sekarang ini dunia berada di era digital. Era dimana hampir setiap aspek di dalam kehidupan manusia sangat berhubungan erat dengan teknologi komputasi. Semakin berkembangnya zaman, manusia terus mengembangkan pengetahuan dan teknologi untuk membantu dan meringankan pekerjaannya. Salah satu bidang penelitian yang sampai saat ini masih berkembang adalah kecerdasan buatan atau yang lebih dikenal dengan sebutan *Artificial Intelligence* (AI).

Pengembangan cabang ilmu pengetahuan *Artificial Intelligence*, salah satunya adalah *computer vision*. *Computer Vision* dapat didefinisikan sebagai disiplin ilmu yang mempelajari tentang bagaimana komputer dapat mengenali objek yang diamati atau diobservasi. Dalam *computer vision* terdapat beberapa permasalahan diantaranya adalah *object detection* dan *image classification*.

*Object detection* (pendeteksian objek) baru-baru ini menjadi salah satu bidang yang paling menarik dalam *computer vision* dan *artificial intelligence* (AI). Pendeteksian objek merupakan teknologi komputer yang berkaitan dengan *computer vision* dan *image processing* yang berhubungan dengan mendeteksi suatu objek dalam citra digital yang dapat berupa warna dan bentuk objek.

Terdapat beberapa metode dalam mendeteksi dan mengenali objek pada sebuah gambar, salah satunya adalah metode *Convolutional Neural Network* (CNN) yang sering digunakan pada data *image*. Penelitian yang dilakukan oleh Imam Taufiq (2018) menggunakan metode *Convolutional Neural Network* dalam melakukan pendeteksian tanda nomor kendaraan bermotor menghasilkan akurasi sebesar 99%. *Convolutional Neural Network* (CNN)

merupakan salah satu metode yang terdapat dalam *deep learning* yang banyak digunakan untuk menyelesaikan permasalahan yang berkaitan dengan *object detection* dan *image classification*. *Convolutional Neural Network* (CNN) banyak digunakan pada penelitian terdahulu karena memiliki tingkat akurasi yang relative tinggi dan memiliki hasil yang signifikan dalam pengenalan citra.

Berdasarkan uraian di atas, dalam penelitian ini akan dibuat sebuah sistem untuk mendeteksi antara meja dan kursi motif ukiran Jepara pada suatu gambar dan video. Adapun algoritma yang digunakan oleh sistem adalah algoritma *Convolutional Neural Network*. Oleh karena itu, peneliti membuat penelitian yang berjudul “**Deep Learning Object Detection pada Video dengan Menggunakan Tensorflow dan Convolutional Neural Network**” dengan studi kasus klasifikasi gambar meja dan kursi motif ukiran Jepara.

Harapannya dengan penelitian ini mampu melestarikan dan mempertahankan budaya ukiran Jepara pada sebuah meja dan kursi untuk mencegah seni ukir Jepara supaya tidak dilupakan oleh masyarakat Indonesia. Harapan lainnya dari penelitian ini yaitu mampu untuk mendeteksi gambar meja dan kursi motif ukiran Jepara dengan baik sehingga nantinya informasi tersebut dapat berguna bagi pihak yang membutuhkannya.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana model yang terbentuk dari hasil pelatihan pada deteksi meja dan kursi motif ukiran Jepara menggunakan *Convolutional Neural Network*?
2. Bagaimana hasil pendeteksian klasifikasi meja dan kursi motif ukiran Jepara pada suatu citra digital?
3. Bagaimana tingkat akurasi pendeteksian meja dan kursi motif ukiran Jepara pada suatu citra digital menggunakan *Convolutional Neural Network*?

### 1.3 Batasan Masalah

Batasan penelitian agar sesuai dengan yang dimaksudkan dan lebih terarah adalah sebagai berikut:

1. *Software* yang digunakan adalah *Python* dengan framework *Tensorflow*.
2. Data yang digunakan dalam penelitian ini merupakan data gambar dengan dua klasifikasi yaitu meja dan kursi dengan motif ukiran Jepara.
3. Dataset gambar diambil melalui *crawling* dari *google image*.
4. Metode yang digunakan adalah *Convolutional Neural Network*.
5. Jumlah *dataset* yang digunakan berjumlah 500 gambar yang terdiri dari:
  - a. *Data Training*  
Data gambar yang digunakan untuk proses *training* berjumlah 470 data gambar dan 470 data label tiap gambar.
  - b. *Data Testing*  
Data gambar yang digunakan untuk proses *testing* berjumlah 30 data gambar dan 30 data label tiap gambar.

### 1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui model yang terbentuk dari hasil pelatihan pada deteksi meja dan kursi motif ukiran Jepara menggunakan *Convolutional Neural Network*.
2. Mengetahui hasil pendeteksian klasifikasi meja dan kursi motif ukiran Jepara pada suatu citra digital.
3. Mengetahui seberapa tinggi tingkat akurasi pendektesian meja dan kursi motif ukiran Jepara pada suatu citra digital.

## 1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Dapat memudahkan suatu pihak dalam mendeteksi meja dan kursi motif ukiran Jepara dengan bantuan teknologi melalui model hasil pengujian penelitian ini.
2. Dengan diketahuinya cara mendeteksi meja dan kursi motif ukiran Jepara pada suatu gambar diharapkan dapat membuat perkembangan pada permasalahan *computer vision* lainnya.
3. Hasil penelitian ini dapat dijadikan acuan untuk penelitian lebih lanjut yang berbasis pada pendeteksian objek pada suatu gambar dan terkait *Convolutional Neural Network*.

## **BAB II**

### **TINJAUAN PUSTAKA**

Sehubungan dengan penelitian yang dilakukan penulis, referensi dari penelitian terdahulu sangat penting untuk dilakukan agar terhindar dari penjiplakan atau duplikasi dari penelitian terdahulu, hal ini bertujuan juga sebagai bahan untuk kontribusi penelitian bagi penulis agar penelitian tentang tema ini terus berkembang. Berikut beberapa ulasan tentang penelitian terdahulu yang pernah dilakukan sebelumnya berkenaan dengan data dan metode yang digunakan.

Penelitian mengenai “Deep Learning untuk Deteksi Tanda Nomor Kendaraan Bermotor Menggunakan Algoritma Convolutional Neural Network Dengan Python dan Tensorflow” yang dilakukan oleh Imam Taufiq (2018). Dalam penelitian ini menggunakan algoritma *Convolutional Neural Network* untuk klasifikasi dan mendeteksi plat kendaraan bermotor pada sebuah gambar. Pada penelitian ini terdapat 502 dataset gambar dan menggunakan perbandingan 80% untuk training serta 20% untuk testing. Proses training membutuhkan lebih dari 25.000 step dengan jumlah *batch* 8 dan pada saat *batch* yang digunakan adalah 4 membutuhkan 100.000 step sampai model yang di *training* mampu mendeteksi keberadaan TNKB serta menghasilkan akurasi sekitar 99% pada sebuah gambar plat kendaraan bermotor.

Penelitian mengenai “*object recognition with deep learning applied to fashion items detection in images*” yang dilakukan oleh Helder Filipe de Sousa Russa (2017). Dalam penelitian ini menggunakan metode Fast R-CNN untuk mengklasifikasikan dan mendeteksi item fashion tertentu yang digunakan oleh orang pada sebuah gambar. Pada penelitian dilakukan 3677 *train image* per kategori dan melakukan 696 *testing image* per kategorinya. Hasil menunjukkan bahwa dengan menggunakan metode CNN untuk mendeteksi fashion item yang dipakai oleh seseorang menghasilkan rata-rata *precision of close* sebesar 78%, untuk *pants* sebesar 65% dan untuk rata-rata aksesoris seperti *glasses* sebesar 57%. Metode Fast R-CNN digunakan untuk lebih mempersingkat waktu dalam pelatihan objek.

Penelitian mengenai “klasifikasi citra menggunakan *convolutional neural network* pada *Caltech 101*” yang dilakukan oleh I Wayan Suartika E.P., dkk (2016). Penelitian ini menggunakan “metode *deep learning* untuk mengklasifikasikan unggas diantaranya dengan kategori emu, flamingo, ibis, pigeon, dan roaster yang terdiri dari 150 citra. Selain kategori tersebut dihasilkan pula 3 kategori yaitu cougar, crocodile, dan face. Hasil dari 5 kategori unggas menunjukkan bahwa persentase keberhasilan 20% sedangkan untuk 3 kategori lainnya menunjukkan persentase keberhasilan 50%. Dengan menggunakan data *training* yang baik dan optimal, maka subset dari data *training* tersebut juga akan menghasilkan klasifikasi yang baik.”

Penelitian mengenai “Deteksi Jenis Kendaraan di Jalan Menggunakan OpenCV” yang dilakukan oleh Alvin Lazaro, dkk (2017). Penelitian ini menggunakan *Haar-like feature* untuk melakukan deteksi objek kendaraan. Tingkat akurasi rata-rata untuk tiga kondisi jalan yang berbeda (sepi, normal, dan padat) adalah sebesar 77.8%, 47.5%, dan 28.2%.

Penelitian mengenai “*Object Detection for Autonomous Driving Using Deep Learning*” yang dilakukan oleh Victor Vaquero Gomez (2015). Penelitian ini menggunakan *Convolutional Neural Network*. Penelitian ini memberikan hasil pengujian pada deteksi objek seseorang pada sebuah video yang menghasilkan akurasi yang cukup tinggi yaitu sebesar 89% dari *Equal Error Rate*.

Penelitian mengenai “Klasifikasi Objek Terdeformasi Berdasarkan Nilai Deviasi Menggunakan Metode Kontinuitas Kontur” yang dilakukan oleh Setiawan Hadi, Akmal, dan Susi Herlina (2014). Penelitian ini bertujuan mengembangkan metode pendeteksian dan pengklasifikasian objek digital. Metode yang dikembangkan adalah metode kontinuitas kontur melalui estimasi klasifikasi objek menggunakan tingkat deviasi. Pengujian yang dilakukan berupa pendeteksian objek lingkaran, elips, DES, persegi, dan garis dengan banyaknya objek sebanyak 60 buah. Penelitian ini menghasilkan tingkat akurasi sebesar 85%.

Penelitian mengenai “Ekstraksi Fitur Untuk Pengenalan Wajah Pada Ras Mongoloid Menggunakan *Principal Component Analysis* (PCA)” yang dilakukan oleh Dwiandi Susantyo. Penelitian ini melakukan beberapa pengujian

menggunakan PCA sebagai pengambilan ciri dari pada citra wajah dan *Euclidean distance* untuk mencari nilai minimum jarak antar citra data training dan citra data test menghasilkan nilai akurasi yang cukup baik sebesar 78,89%, namun membutuhkan data yang banyak sebagai data training.

Berdasarkan penelitian yang disebutkan sebelumnya, diketahui bahwa belum ada penelitian mengenai pendeteksian objek khususnya pada objek meja dan kursi dengan dua klasifikasi menggunakan metode *Convolutional Neural Networks* (CNN). Oleh karena itu, pada penelitian ini akan dilakukan pendeteksian dan pengklasifikasian objek meja dan kursi ukiran Jepara menggunakan *Convolutional Neural Network*.



**Tabel 2.1** Perbandingan dengan Penelitian Terdahulu

No.	Penulis	Judul Penelitian	Metode Penelitian	Hasil
1.	Imam Taufiq (2018)	Deep Learning untuk Deteksi Tanda Nomor Kendaraan Bermotor Menggunakan Algoritma Convolutional Neural Network Dengan Python dan Tensorflow	Algoritma CNN dan Tensorflow	Penelitian ini menggunakan algoritma CNN untuk mendeteksi dan mengklasifikasi tanda kendaraan bermotor pada sebuah motor dengan jumlah 502 dataset, 80% untuk training dan 20% untuk testing. Penelitian ini menghasilkan akurasi sekitar 99% dengan batch 4 dan dengan 100.000 steps sampai proses training berhasil mendeteksi sebuah plat kendaraan bermotor pada sebuah gambar.
2.	Helder Filipe de Sousa Russa (2017)	Computer Vision: Object Recognition with Deep Learning Applied to Fashion Items Detection in Images	<i>Fast R-CNN</i>	Penelitian ini menggunakan 3677 <i>train image</i> per kategori dan 696 <i>testing image</i> per kategorinya. Penelitian ini digunakan untuk mendeteksi fashion item yang dipakai oleh seseorang yang menghasilkan rata-rata <i>precision of close</i> sebesar 78%, untuk <i>pants</i> sebesar 65% dan untuk rata-rata aksesoris seperti <i>glasses</i> sebesar 57%.
3.	I Wayan Suartika E.P., dkk (2016)	Klasifikasi Citra Menggunakan Convolutional Neural Network pada Caltech 101	<i>Convolutional Neural Network</i>	Penelitian ini menunjukkan bahwa metode praproses dan metode klasifikasi dengan metode CNN cukup handal kebenaran dari klasifikasi citra objek. Hal ini terbukti dengan persentase keberhasilan 20% pada kategori 9 unggas sedangkan untuk 3 kategori lainnya menunjukkan persentase

				keberhasilan 50%. Perubahan tingkat <i>confusion</i> tidak mempengaruhi hasil akurasi.
4.	Alvin Lazaro, dkk (2017)	Deteksi Jenis Kendaraan di Jalan Menggunakan OpenCV	Haar-like feature	Tingkat akurasi rata-rata untuk tiga kondisi jalan yang berbeda (sepi, normal, dan padat) adalah sebesar 77.8%, 47.5%, dan 28.2%.
5.	Victor Vaquero Gomez (2015)	<i>Object Detection for Autonomous Driving Using Deep Learning</i>	<i>Convolutional Neural Network</i>	Penelitian ini memberikan hasil pengujian pada deteksi objek seseorang pada sebuah video yang menghasilkan akurasi yang cukup tinggi yaitu sebesar 89% dari <i>Equal Error Rate</i> .
6.	Hadi, Akmal, dan Susi Herlina (2014)	Klasifikasi Objek Terdeformasi Berdasarkan Nilai Deviasi Menggunakan Metode Kontinuitas Kontur	Metode Kontinuitas Kontur	Pengujian yang dilakukan berupa pendeteksian objek lingkaran, elips, DES, persegi, dan garis dengan banyaknya objek sebanyak 60 buah. Penelitian ini menghasilkan tingkat akurasi sebesar 85%.
7.	Dwiandi Susantyo	Ekstraksi Fitur Untuk Pengenalan Wajah Pada Ras Mongoloid Menggunakan <i>Principal Component Analysis</i> (PCA)	<i>Principal Component Analysis</i>	Pengujian pada penelitian ini menggunakan PCA sebagai pengambilan ciri dari pada citra wajah dan <i>Euclidean distance</i> untuk mencari nilai minimum jarak antar citra data training dan citra data test menghasilkan nilai akurasi yang cukup baik sebesar 78,89%, namun membutuhkan data yang banyak sebagai data training.

## **BAB III**

### **LANDASAN TEORI**

#### **3.1 Meja dan Kursi**

Menurut Wikipedia, meja adalah sebuah mebel atau perabotan yang memiliki permukaan datar serta kaki-kaki sebagai penyangga yang bentuk dan fungsinya bermacam-macam. Meja sering digunakan untuk menaruh barang atau makanan. Meja umumnya selalu dipasangkan dengan kursi atau bangku. Secara etimologi, kata meja merupakan serapan dari bahasa Portugis, *mesa* yang berakar dari kata *mensa* dalam bahasa Latin. Dalam bentuknya, meja memiliki bermacam-macam bentuk seperti persegi panjang, persegi, bulat dan elips.

Sedangkan, kursi yaitu salah satu perabotan rumah yang biasa digunakan sebagai tempat duduk. Pada umumnya, kursi memiliki 4 kaki yang digunakan untuk menopang berat tubuh di atasnya. Beberapa jenis kursi, seperti *barstool*, hanya memiliki 1 kaki yang terletak di bagian tengah. Kadang-kadang kursi juga dilengkapi dengan sandaran kaki.

#### **3.2 Ukiran Jepara**

##### **3.2.1 Sejarah**

Sejarah ukir Jepara tidak lepas dari kisah Syeh Muhayat Syah yang biasa disebut Sultan Hadlirin. Ia adalah seorang yang berasal dari Aceh, pada awalnya dia berguru ke negeri Cina daratan dan disana ia dijadikan anak angkat oleh dua orang ahli dalam strategi perang dan mereka adalah kakak beradik yaitu Thae Lin Sing dan Chiwi Guan. Setelah merasa cukup berguru di Cina daratan, Muhayat Syah hendak pulang ke negeri. Akan tetapi dalam perjalanannya, ia bersama 4 kapal yang memuat periuk asli dari Dynasti Yuan terdampar di perairan Jepara. Ketika ia hendak menginjakkan kakinya di bumi Jepara, terdengar sayembara yang konon diselenggarakan oleh Retno Kencono, seorang putri cantik jelita dan pemberani yang sekarang menjadi Ratu dengan julukan Ratu Kalinyamat yang masih dikenang hingga sekarang.

Isi sayembara tersebut adalah “*sopo sing iso ngalahno aku, yen lanang aku bakal suwito sak lawase yen wadon tak daku sedulur sinoro wedi*” yang artinya barang siapa yang bisa mengalahkannya, jika laki-laki akan dijadikannya suami dan jika perempuan akan diangkat menjadi saudara kandungnya. Mendengar ada sayembara, Syeh Muhayat Syah menyamar sebagai orang biasa dan mencoba mengikuti sayembara tersebut. Alhasil, ia memenangkan sayembaranya Retno Kencono dan tidak lama kemudian mereka menikah.

Ketika itu di Cina daratan terjadi perang besar, Thae Lin Sing dan Chiwi Guan kedua orang yang ahli dalam strategi perang tersebut teringat akan anak angkatnya yaitu Sultan Hadlirin, maka dicarilah dia sampai ketemu. Melalui perjalanan yang cukup panjang, akhirnya Thae Lin Sing dan Chiwi Guan mendengar kabar bahwa anak angkatnya berada di Jepara. Merekapun mencarinya dan bertemu di Jepara dengan mengabarkan apa yang sedang terjadi di Cina daratan. Akhirnya demi keselamatan kedua ayah angkatnya Sultan Hadlirin mempersilahkan mereka untuk tinggal di Jepara.

Sultan Hadlirin menyarankan kepada Thae Lin Sing untuk menjadi penasehat perang Sunan Kudus dan Chiwi Guan dipilih menjadi patih di Jepara. Tidak disangka ternyata Chiwi Guan juga ahli dalam memahat di batu putih. Dengan keahlian tersebut dia mendapat gelar Patih Sungging Dhuwung (ahli pahat batu putih). Semenjak lihai memahat akhirnya Badhar Dhuwung sering membuat perabotan rumah tangga dan perkantoran pada jamannya yaitu meja dan kursi bermotif ukiran. Sampai sekarang keahlian dan kebiasaan Patih Sungging menjadi kebiasaan masyarakat Jepara yaitu membuat ukiran pada mebel. Bentuk peninggalan sejarah Patih Sungging adalah motif ukiran batu putih yang sekarang masih melekat pada dinding masjid mantingan (Astana Sultan Hadlirin). Maka dari sanalah Jepara dikenal sebagai kota ukir.

### **3.2.2 Motif Ukiran**

Motif seni ukir asli Jepara memiliki ciri khas sendiri yang dapat terlihat dari motif jumbai atau ujung relung yang daunnya seperti kipas yang sedang terbuka. Pada ujung daun tersebut meruncing dan ada beberapa biji buah yang keluar dari pangkal daun. Selain itu, tangkai relungnya memutar dengan gaya memanjang dan

menjalar membentuk cabang-cabang kecil yang mengisi ruang atau memperindah. Seni ukir Jepara juga dapat diterapkan pada bentuk patung ataupun perabot rumah tangga seperti kursi, meja, dan lainnya.



**Gambar 3.1** Contoh Ukiran pada Kursi



**Gambar 3.2** Contoh Ukiran pada Meja

### 3.3 Pengolahan Citra

#### 3.3.1 Definisi Citra

Citra (*image*) adalah gambar pada bidang dua dimensi dan disusun oleh banyak piksel yang merupakan bagian terkecil dari citra. Pada umumnya, citra dibentuk dari kotak-kotak persegi empat yang teratur sehingga jarak horizontal dan vertikal antara piksel sama pada seluruh bagian citra (Ldya, et al. 2010).

Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat (Sitorus, Syahriol dkk. 2006):

- a. Optik berupa foto,
- b. Analog berupa sinyal video seperti gambar pada monitor televisi,
- c. Digital yang dapat langsung disimpan pada media penyimpanan magnetik.

Citra dapat dikelompokkan menjadi dua bagian yaitu citra diam (*still image*) dan citra bergerak (*moving image*). Citra diam yang ditampilkan secara beruntun (*sekuensial*), sehingga memberi kesan pada mata sebagai gambar yang bergerak. Setiap citra didalam rangkaian tersebut disebut *frame*. Gambar-gambar yang

tampak pada film layar lebar atau televisi pada hakekatnya terdiri dari ratusan sampai ribuan *frame*. (Sitorus, Syahriol dkk. 2006)

### 3.3.2 Definisi Citra Digital

Citra digital merupakan suatu matriks dimana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya yang disebut sebagai elemen gambar atau piksel menyatakan nilai tingkat derajat keabuan pada titik tersebut. Citra digital berbentuk matriks dengan ukuran M (baris/tinggi) x N (kolom/lebar) yang akan tersusun sebagai berikut:

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0, m-1) \\ f(1,0) & f(1,1) & \dots & f(1, m-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1, M-1) \end{bmatrix} \quad (3.1)$$

Suatu citra  $f(x, y)$  dalam fungsi matematis dapat dituliskan sebagai berikut:

$$0 \leq x \leq M - 1$$

$$0 \leq y \leq N - 1$$

$$0 \leq f(x, y) \leq G - 1$$

Dimana: M=banyaknya baris pada *array* citra

N=banyaknya kolom pada *array* citra

G=banyaknya skala keabuan (*graylevel*)

Proses perubahan citra analog menjadi citra digital dinamakan dengan digitasi. Digitasi adalah proses mengubah gambar, teks, atau suara dari benda yang dapat dilihat ke dalam data elektronik dan dapat disimpan serta diproses untuk keperluan lainnya. Citra digital pada komputer dipetakan menjadi bentuk grid atau elemen piksel berbentuk matriks 2 dimensi. Setiap piksel-piksel tersebut memiliki angka yang mempresentasikan channel warna. Angka pada setiap piksel disimpan secara berurutan oleh komputer dan sering dikurangi untuk keperluan kompresi maupun pengolahan tertentu.

### 3.3.3 Tipe Citra Digital

Citra digital dapat dikategorikan dalam beberapa jenis, yaitu:

#### 1. Citra Biner

Citra biner (*binary image*) adalah citra yang hanya mempunyai nilai derajat keabuan yaitu hitam dan putih. Piksel-piksel objek bernilai 1 dan

piksel-piksel latar belakang bernilai 0. Piksel yang bernilai 0 melambangkan warna putih dan piksel yang bernilai 1 melambangkan warna hitam pada saat menampilkan citra.

## 2. Citra Keabuan

Citra keabuan (*greyscale*) adalah citra yang disetiap pikselnya mengandung satu *layer* dimana nilai intensitasnya berada pada nilai 0 (hitam) – 255 (putih).

## 3. Citra Warna

Citra warna (RGB) adalah citra digital yang memiliki informasi warna pada setiap pikselnya. Sistem pewarnaan citra warna ada beberapa macam seperti RGB, CMYK, HSV, dll. RGB adalah model warna yang terdiri dari merah, hijau, dan biru yang digabungkan dalam membentuk suatu susunan warna yang luas. Untuk monitor komputer, nilai rentangnya paling kecil = 0 dan paling besar = 255.

### 3.3.4 Ekstraksi Ciri Suatu Gambar

(Marques dan Furht 2002): Ekstraksi ciri merupakan proses pengingkdeksan suatu database citra beserta isinya. Secara matematik, setiap ekstraksi ciri merupakan *encode* dari vektor  $n$  dimensi yang disebut dengan vektor ciri. Komponen vektor ciri dihitung dengan pemrosesan citra dan teknik analisis serta digunakan untuk membandingkan citra yang satu dengan citra yang lain. Ekstraksi ciri diklasifikasikan menjadi 3 jenis, diantaranya *low level*, *middle level*, dan *high level*. *Low level feature* merupakan ekstraksi ciri berdasarkan isi visual seperti warna dan tekstur. *Middle level feature* merupakan ekstraksi berdasarkan wilayah citra yang ditentukan dengan segmentasi, sedangkan *high level feature* merupakan ekstraksi ciri berdasarkan informasi semantik yang terkandung dalam citra, macam-macam ekstraksi ciri:

#### a. Warna

Warna merupakan salah satu ciri visual yang digunakan dalam *Content Based Image Retrieval (CBIR)*. Warna sangat baik jika digunakan untuk temu kembali citra karena memiliki hubungan yang sangat kuat dengan

objek dalam sebuah citra, yang melatar belakangi gabungan background, skala, orientasi, perspektif dan ukuran.

b. Bentuk

Bentuk merupakan ciri dalam suatu citra yang sangat esensial untuk segmentasi citra karena dapat mendeteksi objek atau batas suatu wilayah. Proses yang dapat digunakan untuk menentukan ciri bentuk adalah deteksi tepi, *threshold*, segmentasi, dan perhitungan moment seperti (mean, median dan standard deviasi dari setiap lokal citra).

c. Tekstur

Tekstur merupakan ciri intrinsic dari suatu citra yang terkait tingkat kekasaran (*roughness*), granularitas (*granulation*), dan keteraturan (*regularity*) susunan piksel. Aspek tekstural dari sebuah citra dapat dimanfaatkan sebagai dasar dari segmentasi, klasifikasi, maupun interpretasi citra.

### 3.4 *Web Crawler*

*Web Crawler* atau juga disebut *web spider* memiliki tugas untuk men-*crawl* (merayapi) seluruh informasi dari suatu *website*. *Crawling* akan menggali seluruh data dari suatu *website* termasuk didalamnya komponen *website* seperti: meta data, *keywords* dan lain sebagainya. *Web Crawler* dimulai dengan me-list daftar URL yang akan dikunjungi, yang disebut dengan seed. *Web Crawler* akan mengunjungi URL yang ada di daftar dan mengidentifikasi semua *hyperlink* di halaman tersebut serta menambahkannya ke dalam daftar URL yang akan dikunjungi yang disebut *crawl frontier*. URL yang telah ada dikunjungi dan diambil informasi sesuai yang dibutuhkan. Dengan banyaknya jumlah URL yang mungkin di *crawl* oleh *crawler server* yang membuatnya sulit untuk menghindari pengambilan konten yang sama. Misalkan protokol HTTP GET membuat kombinasi URL yang sangat banyak dan sedikit dari URL tersebut menghasilkan konten yang berbeda dan selebihnya menghasilkan konten yang sama untuk URL yang berbeda dan selebihnya menghasilkan konten yang sama untuk URL yang berbeda, inilah yang menimbulkan masalah bagi *crawler* agar bisa mengambil konten yang berbeda dari URL-URL tersebut.



### 3.5 *Artificial Intelligence* (Kecerdasan Buatan)

*Artificial Intelligence* merupakan ilmu dan teknik pembuatan mesin cerdas, khususnya program komputer cerdas. Hal ini terkait dengan tugas yang sama dengan menggunakan komputer untuk memahami kecerdasan manusia, tetapi *Artificial Intelligence* tidak harus membatasi dirinya terhadap metode yang diamati secara biologis (McCarthy, 2007:2).

Menurut pengertian Dobrev (2004:2) yang mengatakan bahwa *Artificial Intelligence* sebagai pembelajaran bagaimana membuat komputer melakukan hal-hal yang dimana saat ini masih lebih baik dilakukan oleh manusia.

### 3.6 *Computer Vision*

*Computer Vision* merupakan salah satu cabang ilmu pengetahuan yang bertujuan untuk membuat suatu keputusan yang berguna untuk mengenali objek fisik nyata dan keadaan berdasarkan sebuah gambar atau citra (Shapiro & Stockman, 2001). *Computer Vision* menjadikan komputer “*acts like human sight*”, sehingga mendekati kemampuan manusia dalam menangkap informasi visual. Kemampuan itu diantaranya adalah:

- *Object Detection*: Mengenali sebuah objek ada pada *scene* dan mengetahui dimana batasannya.
- *Recognition*: Menempatkan label pada objek.
- *Description*: Menugaskan properti kepada objek.
- *3D Inference*: Menafsirkan adegan 3D dari 2D yang dilihat.
- *Interpreting motion*: Menafsirkan gerakan.

### 3.7 *Machine Learning*

*Machine Learning* merupakan cabang ilmu dari *Artificial Intelligence* yang memungkinkan komputer memiliki kemampuan untuk belajar tanpa perlu di program lagi (Arthur Samuel. 1959). Secara sederhana *machine learning* membangun sebuah algoritma yang memungkinkan program komputer untuk belajar dan melakukan tugasnya sendiri tanpa adanya instruksi dari penggunanya. Algoritma semacam ini bekerja dengan cara membangun sebuah model dari *input* atau masukan untuk dapat menghasilkan suatu prediksi atau pengambilan keputusan berdasarkan data yang ada. *Machine learning* berhubungan dengan

*computational statistics* yang berfokus pada suatu prediksi atau pembuatan keputusan berdasarkan penggunaan komputer. Beberapa implementasi dari *machine learning* adalah *text analysis*, *image processing*, *fincance*, *search and recommendation engine*, *speech understanding*.

Dalam pembelajaran *machine learning*, terdapat tiga kategori utama yaitu:

a. *Supervised Learning*

Pada *supervised learning*, data yang dimiliki dilengkapi dengan label/kelas yang menunjukkan klasifikasi atau kelompok data tersebut berada. Model yang dihasilkan adalah model prediksi dari data yang telah diberi label.

b. *Unsupervised Learning*

Pada *unsupervised learning*, data pembelajaran tidak memiliki label/kelas sehingga harus mencari struktur dari data yang ada, kemudian melakukan pengelompokan berdasarkan informasi yang dimiliki.

c. *Reinforcement Learning*

Pada *reinforcement learning*, pembelajaran terhadap apa yang akan dilakukan (bagaimana memetakan situasi ke dalam aksi) untuk mendapatkan *reward* yang maksimal. Pembelajar tidak diberitahu aksi mana yang akan diambil, tetapi lebih pada menemukan aksi mana yang dapat memberikan *reward* maksimal dengan mencoba menjalankannya.

### 3.8 *Deep Learning*

*Deep learning* adalah salah satu bidang *machine learning* yang memanfaatkan banyak *layer* pengolahan informasi nonlinier untuk melakukan ekstraksi fitur, pengenalan pola, dan klasifikasi (Deng dan Yu, 2014). Menurut Goodfellow, dkk. (2016), *deep learning* adalah sebuah pendekatan dalam penyelesaian masalah pada sistem pembelajaran komputer yang menggunakan konsep hierarki. Konsep hierarki membuat komputer mampu mempelajari konsep yang kompleks dengan menggabungkan dari konsep-konsep yang lebih sederhana. Jika digambarkan sebuah graf bagaimana konsep tersebut dibangun di atas konsep yang lain, graf ini akan dalam dengan banyak *layer*, hal tersebut menjadi alasan disebut sebagai *deep learning* (pembelajaran mendalam).

### 3.9 *Image Processing (Pengolahan Citra)*

Menurut Murni, pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran berbentuk sebuah citra. Teknik pengolahan citra menggunakan komputer untuk mendigitasi pola bayangan dan warna pada gambar yang sudah tersedia. Informasi yang telah terdigitasi kemudian ditransfer ke layar monitor video. Pengolahan citra banyak digunakan dalam dunia fotografi misalnya mengubah intensitas cahaya sebuah foto atau dalam dunia perfilman misalnya animasi, dunia kedokteran misalnya membuat analisa medis, dan pada dunia game.

### 3.10 *Segmentasi*

Menurut Ramesh Jain, Kasturi, dan Schunk (1995), segmentasi merupakan sebuah metode untuk membagi sebuah gambar menjadi sub-sub gambar yang disebut area (region). Ada 2 pendekatan yang dapat digunakan untuk membagi gambar-gambar menjadi daerah-daerah tertentu yaitu *Region-based* dan *Boundary estimation* menggunakan *edge detection*.

Pada pendekatan *region-based*, semua piksel yang berkorespondensi dengan sebuah objek dikelompokkan bersama dan diberikan *flag* yang menandakan bahwa mereka merupakan satu area. Dua prinsip yang penting dalam pendekatan ini adalah *value similarity* dan *spatial proximity*. Dua piksel dapat dikelompokkan menjadi satu bila mempunyai karakteristik intensitas yang serupa atau bila keduanya memiliki jarak yang berdekatan. Sedangkan pada pendekatan *boundary estimation* menggunakan *edge detection*, segmentasi dilakukan dengan menemukan piksel-piksel yang terletak pada sebuah batas are. Piksel tersebut (atau yang disebut sebagai *edges*) dapat diperoleh dengan melihat piksel-piksel yang berdekatan (*neighbouring pixels*).

### 3.11 *Pattern Recognition*

*Pattern Recognition* atau pengenalan pola juga disebut pembelajaran mesin yang mempelajari berbagai teknik matematika seperti teknik statistika, jaringan syaraf tiruan, mesin vektor pendukung, dan lain-lain untuk mengklasifikasikan pola yang berbeda. Data *input* untuk *pattern recognition* dapat berupa data. Teknik pengenalan pola banyak digunakan pada *computer vision*.

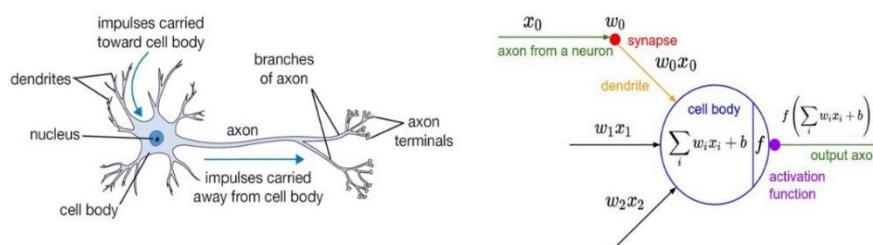
Pengenalan pola memiliki arti bidang studi yang melakukan proses analisis gambar yang bentuk masukannya adalah gambar itu sendiri atau dapat berupa citra digital dan bentuk keluarannya yaitu suatu deskripsi (Murni, 1992). Tujuan dari adanya pengenalan pola adalah meniru kemampuan manusia dalam mengenali suatu objek atau pola tertentu.

### 3.12 Object Detection

*Object detection* menentukan keberadaan suatu objek dan ruang lingkungnya serta lokasi pada sebuah gambar. Hal ini dapat diperlakukan sebagai pengenalan objek kelas dua, dimana satu kelas mewakili kelas objek dan kelas lain mewakili kelas non-objek. Deteksi objek dapat dibagi lagi menjadi *soft detection* dan *hard detection*. *Soft detection* hanya mendeteksi adanya objek sedangkan *hard detection* mendeteksi adanya objek serta lokasi objek (Jalled, 2016).

### 3.13 Artificial Neural Network

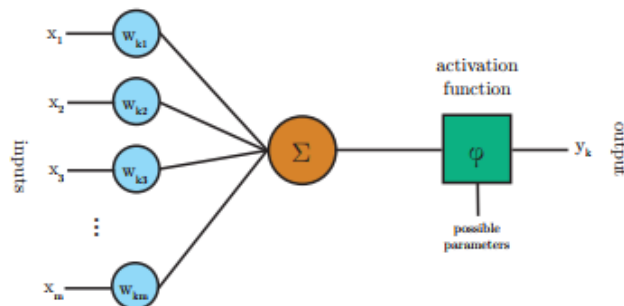
*Neural Network* merupakan suatu metode *artificial intelligence* yang konsepnya meniru sistem jaringan syaraf yang ada pada tubuh manusia, dimana dibangun *node-node* yang saling berhubungan satu dengan yang lainnya. *Node* tersebut terhubung melalui suatu *link* yang biasa disebut dengan istilah *weight* atau bobot. *Neural network* pertama kali dirancang oleh *Warren McCulloch* dan *Walter Pitts* pada tahun 1943 yang dikenal dengan *McCulloch-Pitts neurons* tentang dua neuron aktif secara bersamaan kemudian kekuatan tersebut terkoneksi antara neuron yang seharusnya bertambah. Kemudian pada tahun 1957, Frank Rosenblatt mengenalkan dan mengembangkan sekumpulan besar jaringan saraf tiruan yang disebut *perceptrons*.



**Gambar 3.3** Ilustrasi Neuron dengan Model Matematisnya

Menurut Haykin (2009), *neuron* didefinisikan sebagai unit pengolah informasi yang merupakan dasar dari proses sebuah jaringan saraf tiruan. Untuk

mendetailkan sebuah operasi yang dikerjakan oleh sebuah neuron atau *perceptron* dapat dilihat pada gambar berikut ini.



**Gambar 3.4** *Artificial Neuron*

Pada gambar dijelaskan ada tiga elemen dasar dari model saraf yaitu:

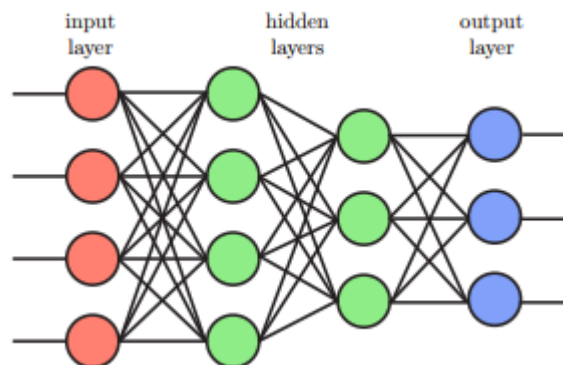
1. Satu set dari sinapsis, atau penghubung yang masing-masing digolongkan oleh bobot atau kekuatannya.
2. Sebuah penambah untuk menjumlahkan sinyal-sinyal *input*. Ditimbang dari kekuatan sinaptik masing-masing neuron.
3. Sebuah fungsi aktivasi untuk membatasi amplitudo *output* dari neuron. Fungsi ini bertujuan membatasi jarak amplitude yang diperbolehkan oleh sinyal *output* menjadi sebuah angka yang terbatas.

*Neural* pada gambar di atas disimbolkan sebagai  $y_k$  yang menerima  $m$  buah *input*  $x_1, \dots, x_m \in \mathbb{R}$ , dapat berasal dari data ataupun *output* dari *layer* sebelumnya. Variabel yang tidak dianggap sebagai *input* (atau sebagai *dummy*) dan selalu bernilai satu dinamakan sebagai bias. Operasi pada sebuah *perceptron* merupakan dua buah operasi terpisah yaitu operasi linear atau *sum* dan operasi *non-linear* atau aktivasi yang dapat dilihat dari persamaan berikut:

$$y_k = \varphi(s_k) \quad (3.1)$$

$$y_k = \varphi\left(\sum_{j=0}^m w_{kj}x_j\right) \quad (3.2)$$

### 3.13.1 Multilayer Networks



**Gambar 3.5** Multilayer Neural Network

*Multilayer Neural Network* adalah *neural network* yang memiliki karakteristik *multi layer* dimana setiap *node* pada suatu *layer* terhubung dengan setiap *node* pada *layer* di depannya. Arsitektur umpan maju (*feed forward network*) menggunakan metode *supervised learning* yang dibedakan dengan adanya keberadaan satu atau lebih *hidden layer*. *Hidden* berarti bagian dari *neural network* secara langsung tidak terlihat oleh *input* atau *output* dari jaringan tersebut. Fungsi dari *hidden layer* adalah untuk mengintervensi antara *input* eksternal dan *output* dari jaringan dengan menambah satu atau lebih *hidden layer*, jaringan dapat mengeluarkan statistik tingkat tinggi dari *input*.

Sumber node di *input layer* dari jaringan menyediakan masing-masing elemen dari pola aktivasi (*vector input*), yang merupakan sinyal *input* yang diaplikasikan ke neuron-neuron pada *layer* kedua (*hidden layer* pertama). Sinyal *output* dari *layer* kedua digunakan sebagai *input-input* ke *layer* ketiga, dan seterusnya sampai ke sisa dari jaringan.

Umumnya operasi model jaringan ini terdapat dua mekanisme kerja yaitu:

a. Mekanisme latihan (*training*)

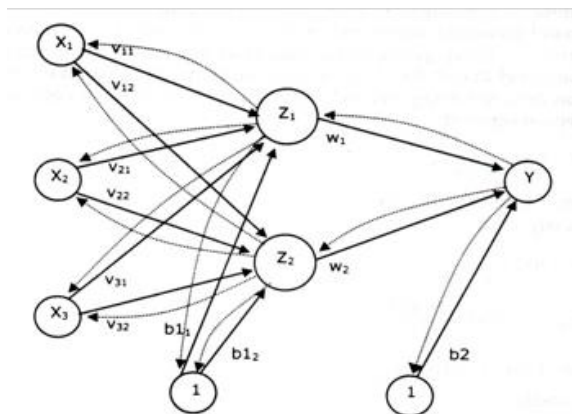
Pada saat *training*, jaringan dilatih untuk dapat menghasilkan data sesuai dengan target yang diharapkan melalui satu atau lebih pasangan data (data *input* dan data target). Semakin lama waktu latihan yang diberikan maka kinerja jaringan akan semakin baik.

b. Mekanisme pengujian (*testing*)

Pada mekanisme ini, jaringan diuji supaya dapat mengenali sesuai dengan yang diharapkan setelah melalui proses latihan yang diberikan.

### 3.13.2 Backpropagation

Salah satu metode yang digunakan dalam *neural network* dan yang paling sering digunakan dalam berbagai bidang aplikasi, seperti pengenalan pola, peramalan dan optimisasi adalah *backpropagation* karena metode ini menggunakan pembelajaran yang terbimbing. Algoritma ini menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Dalam mendapatkan *error* ini, pertama harus melakukan *forward propagation*.



**Gambar 3.6** Backpropagation

Arsitektur jaringan *backpropagation* seperti yang ditunjukkan pada Gambar 3.6 menunjukkan bahwa *neural network* terdiri dari tiga unit *neuron* pada *input layer* yaitu  $x_1$ ,  $x_2$ , dan  $x_3$ , dua *hidden layer* yaitu  $z_1$  dan  $z_2$ , serta 1 unit *neuron* pada *output layer*. Terdapat bobot yang menghubungkan *neuron* pada *input layer* dengan *neuron* pertama pada lapisan tersembunyi yaitu  $v_{11}$ ,  $v_{21}$ ,  $v_{31}$ . Bobot  $b_{11}$  dan  $b_{12}$  merupakan bobot bias yang menuju *neuron* pertama dan kedua pada lapisan tersembunyi. Bobot yang menghubungkan  $z_1$  dan  $z_2$  dengan *neuron* pada lapisan *output* adalah  $w_1$  dan  $w_2$ . Bobot bias  $b_2$  menghubungkan lapisan tersembunyi dengan lapisan *output*.

### 3.13.3 Tipe Fungsi Aktivasi

Beberapa fungsi aktivasi yang dipakai dalam *neural network* adalah:

1. Fungsi *sigmoid biner* (*logsig*)

Fungsi ini umumnya digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode *backpropagation* Fungsi *sigmoid biner* memiliki nilai antara 0 sampai 1. Fungsi *sigmoid biner* dirumuskan sebagai:

$$\varphi(s) = \frac{1}{1+e^{-x}} \quad (3.3)$$

## 2. Fungsi *sigmoid bipolar (tansig)*

Fungsi ini memiliki *range* antara 1 sampai -1. Fungsi *sigmoid bipolar* dirumuskan sebagai:

$$\varphi(s) = \frac{1-e^{-x}}{1+e^{-x}} \quad (3.4)$$

### 3.14 *Learning Rate*

Penggunaan parameter *learning rate* memiliki pengaruh penting terhadap waktu yang dibutuhkan untuk tercapainya target yang diinginkan. Secara perlahan akan mengoptimalkan nilai perubahan bobot dan akan menghasilkan *error* yang lebih kecil (Fajri, 2011). Variabel yang terdapat *learning rate* menyatakan suatu konstanta yang bernilai antara 0.1 sampai 0.9. Nilai tersebut menunjukkan pada kecepatan belajar dari jaringan. Jika nilai *learning rate* nya terlalu kecil maka *epoch* yang dibutuhkan juga semakin banyak untuk mencapai nilai target yang diinginkan, sehingga menyebabkan proses *training* membutuhkan waktu yang lama. Semakin besar nilai *learning rate* yang digunakan maka proses pelatihan jaringan akan semakin cepat, namun jika terlalu besar akan mengakibatkan jaringan tersebut tidak stabil dan menyebabkan nilai *error* berulang pada nilai tertentu, sehingga mencegah *error* mencapai target yang diharapkan. Pada pemilihan nilai variabel *learning rate* diharapkan harus optimal agar didapatkan proses *training* yang cepat (Hermawan, 2006).

### 3.15 *ReLU (Rectrified Linear Unit)*

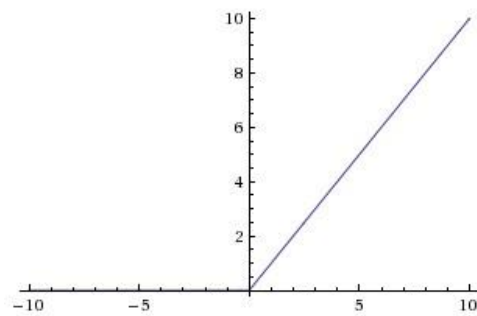
ReLU merupakan fungsi aktivasi pada *Artificial Neural Network* yang saat ini banyak digunakan, berikut rumus pada ReLU.

$$f(x) = \max(x, 0) \quad (3.5)$$

Dimana  $x$  merupakan *input neuron* yang dikenal sebagai fungsi *ramp* dan analog dengan rektifikasi *half-wave* pada teknik elektro. Jika *input* lebih besar 0, *outputnya* sama dengan *input*. Fungsi ReLU lebih mirip *neuron* seperti pada tubuh



manusia. Aktivasi ReLu pada dasarnya sebuah fungsi aktivasi non-linier yang paling sederhana. Bila mendapatkan *input* positif, turunannya hanya 1, dengan kata lain, aktivasi hanya men-threshold pada nilai nol. Penelitian menunjukkan bahwa ReLu menghasilkan pelatihan yang lebih cepat untuk jaringan besar.

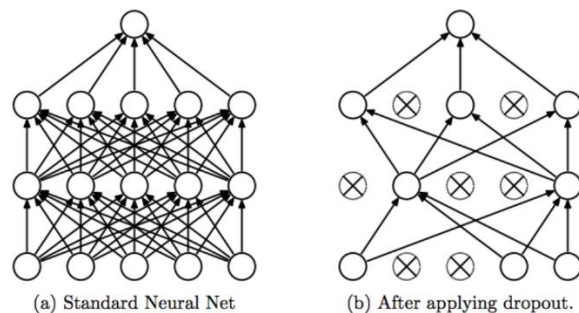


**Gambar 3.7** Grafik Fungsi Aktivasi ReLU

### 3.16 Dropout Regularization

*Regularization* merupakan teknik yang digunakan untuk mengurangi *overfitting*, yakni kondisi dimana sistem jaringan syaraf tiruan mampu belajar dengan baik dengan data pelatihan, namun tidak bisa menggeneralisasi pada data tes. Terdapat beberapa teknik dalam regularisasi, misalnya L2 *regularization* dan *dropout*.

*Dropout* merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses *learning*. *Dropout* mengacu kepada menghilangkan *neuron* yang berupa *hidden* maupun *layer* yang *visible* di dalam jaringan. Dengan menghilangkan suatu *neuron*, berarti menghilangkannya sementara dari jaringan yang ada. *Neuron* yang akan dihilangkan akan dipilih secara acak. Setiap *neuron* akan diberikan probabilitas  $p$  yang bernilai antara 0 dan 1.0. Berikut adalah contoh *Neural Network* sebelum dan sesudah adanya proses *dropout*.



**Gambar 3.8** Before and After Dropout

### 3.17 Convolutional Neural Network

*Convolutional network* atau yang dikenal dengan *convolutional neural network* (CNN) adalah tipe khusus dari *neural network* untuk memproses data yang mempunyai topologi jala atau *grid-like topology*. Pemberian nama *convolutional neural network* mengindikasikan bahwa jaringan tersebut menggunakan operasi matematika yang disebut konvolusi. Konvolusi sendiri adalah sebuah operasi *linear*. Jadi *convolutional network* adalah *neural network* yang menggunakan konvolusi minimal pada salah satu lapisannya (LeCun et al., 2015). *Convolutional neural network* (ConvNets) merupakan *special case* dari *artificial neural network* (ANN) yang saat ini diklaim sebagai model terbaik untuk memecahkan masalah *object recognition* dan *detection*.

*Convolutional Neural Network* (CNN) termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Secara teknis, *convolutional network* adalah arsitektur yang bisa di *training* dan terdiri dari beberapa tahap. *Input* dan *output* dari masing-masing tahap adalah beberapa *array* yang disebut *feature map* atau peta fitur. *Output* dari masing-masing tahap adalah *feature map* hasil pengolahan dari semua lokasi pada *input*. Masing-masing tahap terdiri dari tiga *layer* yaitu *convolution layer*, *activation layer* dan *pooling layer*.

#### 3.17.1 Convolution Layer (Conv. Layer)

*Convolutional Layer* merupakan *layer* pertama yang menerima *input* gambar langsung pada arsitektur. Operasi pada *layer* ini sama dengan operasi konvolusi yaitu melakukan operasi kombinasi linier *filter* terhadap daerah lokal. *Filter* merupakan representasi bidang reseptif dari *neuron* yang terhubung ke dalam daerah lokal pada *input* gambar. *Convolutional layer* melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input*. Secara umum operasi konvolusi dapat ditulis dengan rumus berikut.

$$s(t) = (x * w)(t) \quad (3.5)$$

Pada fungsi  $s(t)$  menghasilkan *output* tunggal yaitu *Feature Map*, argumen pertama berupa *input* yang merupakan  $x$  dan argument kedua yang merupakan  $w$  sebagai kernel atau *filter*. Jika melihat *input* sebagai citra dua dimensi, maka  $(t)$  bisa diasumsikan sebagai sebuah piksel dan menggantinya dengan  $i$  dan  $j$ . Oleh karena itu, untuk operasi konvolusi dengan lebih dari satu dimensi dapat digunakan rumus berikut.

$$S_{(i,j)} = (K * I)_{(i,j)} = \sum \sum I_{(i-m,j-n)} K_{(m,n)} \quad (3.6)$$

Pada persamaan 3.6 merupakan perhitungan dalam operasi konvolusi dengan  $i$  dan  $j$  sebagai piksel dari sebuah citra. Perhitungannya bersifat komutatif dan muncul ketika  $K$  sebagai kernelnya serta  $I$  sebagai *input* dan kernel yang dapat dibalik relative terhadap *input*. Operasi konvolusi dapat dilihat sebagai perkalian matriks antara citra *input* dan *filter* dimana *output*nya dapat dihitung dengan *dot product* (Rismiyati, 2016).

### 3.17.2 *Stride*

*Stride* adalah parameter yang menentukan berapa jumlah pergeseran *filter*. Jika nilai *stride* adalah satu, maka *feature map* akan bergeser sebanyak 1 *pixels* secara horizontal atau vertikal. Semakin kecil *stride* yang digunakan, maka semakin *detail* informasi yang didapatkan dari sebuah *input*, namun membutuhkan komputasi lebih jika dibandingkan dengan *stride* yang besar.

### 3.17.3 *Padding*

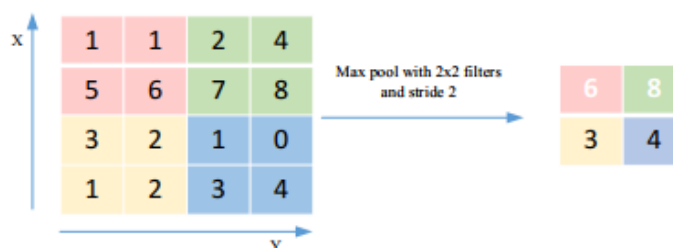
*Padding* atau *Zero Padding* adalah parameter yang menentukan jumlah *pixels* (berisi nilai nol) yang akan ditambahkan di setiap sisi dari *input*. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi *output* dari *feature map*. Penggunaan *padding* dapat untuk mengatur dimensi *output* agar tetap sama seperti dimensi *input* atau setidaknya tidak berkurang drastis sehingga dapat dilakukan ekstraksi *feature* yang lebih mendalam.

### 3.17.4 *Crossentropy Loss Function*

*Loss function* merupakan fungsi yang menggambarkan kerugian yang terkait dengan semua kemungkinan yang dihasilkan oleh sebuah model. *Loss function* ini bekerja ketika model pembelajaran memberikan kesalahan yang harus diperhatikan. *Loss function* yang baik memberikan nilai *error* yang rendah.

### 3.17.5 Pooling Layer

*Pooling* atau *subsampling* adalah pengurangan ukuran matriks dengan menggunakan operasi *pooling*. *Pooling layer* biasanya dilakukan setelah *conv*. Layer. Terdapat dua macam *pooling* yang sering dipakai yaitu *average pooling* dan *max pooling*. Dalam *average pooling*, nilai yang diambil adalah nilai rata-rata, sementara pada *max pooling*, nilai yang diambil adalah nilai maksimal.



**Gambar 3.9** Proses *Max Pooling*

Pada Gambar 3.9 menunjukkan adanya operasi *max-pooling* untuk citra berukuran 4x4 dengan *pooling mask* yang berukuran 2x2. Output dari proses *pooling* adalah matriks dengan dimensi yang lebih kecil dibanding dengan matriks awal. Proses konvolusi dan *pooling* dilakukan beberapa kali sehingga didapatkan peta fitur dengan ukuran yang dikehendaki. Peta fitur tersebut akan menjadi *input* bagi *fully connected neural network* (sumber: medium.com).

### 3.17.6 Activation Function

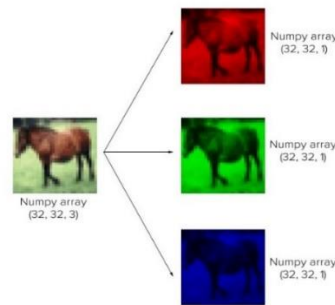
*Activation function* merupakan sebuah *node* yang ditambahkan di akhir *output* dari setiap jaringan syaraf. *Activation function* juga dikenal sebagai *Transfer Function* yang digunakan untuk menentukan *output neural network*. *Activation function* dibagi menjadi dua tipe yaitu linier dan non linier (Sharma, 2017). Pada arsitektur CNN, fungsi aktivasi terletak pada perhitungan akhir keluaran *feature map* atau sesudah proses perhitungan konvolusi atau pooling untuk menghasilkan suatu pola fitur. Beberapa macam fungsi aktivasi yang sering digunakan dalam penelitian antara lain fungsi sigmoid, tanh, *Rectified Linear Unit* (ReLU), Leaky ReLU (LReLU) dan *Parametric ReLU*.

### 3.17.7 Arsitektur Jaringan CNN

Arsitektur dari CNN dibagi menjadi 2 bagian besar, yaitu:

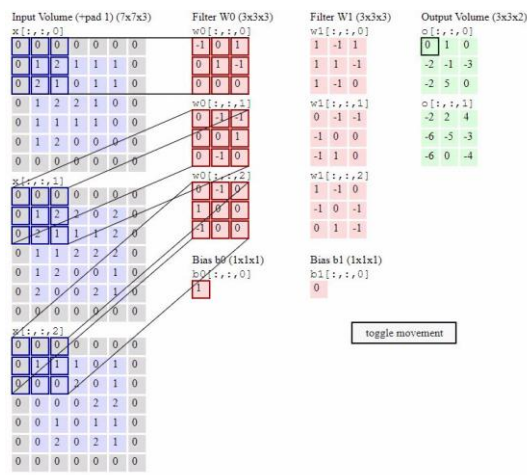
- a. *Feature Extraction Layer*

Proses yang terjadi pada arsitektur ini adalah melakukan *encoding* dari sebuah *image* menjadi *features* yang berupa angka-angka yang mempresentasikan *image* tersebut atau *feature extraction*.



**Gambar 3.10** *Image RGB*

Pada Gambar 3.10 merupakan *channel RGB (Red, Green, Blue) image* berukuran  $32 \times 32$  *pixels* yang sebenarnya adalah multidimensional *array* dengan ukuran  $32 \times 32 \times 3$  (3 merupakan jumlah *channel*). *Convolutional layer* terdiri dari *neuron* yang tersusun sehingga membentuk sebuah filter dengan panjang dan tinggi (*pixels*). Sebagai contoh, pada layer pertama pada *feature extraction layer* biasanya adalah *conv. layer* dengan ukuran  $5 \times 5 \times 3$ . Panjang 5 *pixels*, tinggi 5 *pixels* dan tebal/jumlah nya 3 buah sesuai dengan *channel* dari *image* tersebut. Ketiga filter ini akan digeser keseluruhan bagian dari gambar. Setiap pergeseran akan dilakukan operasi “dot” antara *input* dan nilai dari filter tersebut sehingga menghasilkan sebuah *output* atau biasa disebut dengan *activation map* atau *feature map*.



**Gambar 3.11** *Feature Map*

### b. *Fully Connected Layer*

*Feature map* yang dihasilkan dari *feature extraction layer* masih berbentuk *multidimensional array*, sehingga harus melakukan “flatten” atau *reshape feature map* menjadi sebuah *vector* agar bisa digunakan sebagai *input* dari *fully-connected layer*. *Fully Connected Layer* yang dimaksud disini adalah *Multi Layer Perceptron* yang sudah pernah dipelajari.

Lapisan *Fully-Connected* adalah lapisan di mana semua *neuron* aktivasi dari lapisan sebelumnya terhubung semua dengan *neuron* di lapisan selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua *neuron* di lapisan. Lapisan *Fully-Connected* biasanya digunakan pada metode *Multi Layer Perceptron* untuk mengolah data sehingga bisa diklasifikasikan. Perbedaan antara lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah *neuron* di lapisan konvolusi terhubung hanya ke daerah tertentu pada *input*, sementara lapisan *Fully-Connected* memiliki *neuron* yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoperasikan produk dot, sehingga fungsinya tidak begitu berbeda (Danukusumo, 2017).

### 3.18 *Python*

*Python* merupakan bahasa pemrograman dengan tujuan umum yang dikembangkan secara khusus untuk membuat *source code* mudah dibaca. *Python* juga memiliki *library* yang lengkap sehingga memungkinkan *programmer* untuk membuat aplikasi yang mutakhir dengan menggunakan *source code* yang tampak sederhana (Ljubomir Perkovic, 2012).

### 3.19 *Tensorflow*

*Tensorflow* merupakan perpustakaan perangkat lunak yang dikembangkan oleh Tim Google Brain dalam organisasi penelitian Mesin Cerdas Google, untuk tujuan melakukan pembelajaran mesin dan penelitian jaringan syaraf dalam. *Tensorflow* menggabungkan aljabar komputasi teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan. Fitur utamanya meliputi:

1. Mendefinisikan, mengoptimalkan, dan menghitung secara efisien ekspresi matematis yang melibatkan *array multidimension* (tensors).
2. Pemrograman pendukung jaringan syaraf dalam dan teknik pembelajaran mesin.
3. Penggunaan GPU yang transparan, mengotomatisasi manajemen dan optimalisasi memori yang sama dan data yang digunakan. *Tensorflow* bisa menulis kode yang sama dan menjalankannya baik di CPU atau GPU. Lebih khususnya lagi, *Tensorflow* akan mengetahui bagian perhitungan yang harus dipindahkan ke GPU.
4. Skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

## BAB IV

### METODOLOGI PENELITIAN

#### 4.1 Populasi dan Sampel Penelitian

Populasi yang digunakan dalam penelitian ini adalah seluruh gambar meja dan kursi ukiran Jepara sedangkan sampel pada penelitian ini adalah gambar meja dan kursi ukiran Jepara dengan total 500 gambar.

#### 4.2 Variabel dan Definisi Operasional Penelitian

Variabel yang digunakan dalam penelitian ini ditampilkan dalam Tabel 4.1 tentang penjelasan dan definisi operasional penelitian:

**Tabel 4.1** Definisi Operasional Variabel

Variabel	Definisi Operasional Variabel
Meja	Gambar yang berisi meja dengan ukiran Jepara
Kursi	Gambar yang berisi kursi dengan ukiran Jepara

Data gambar dibagi kedalam 2 kelompok yaitu data *train* dan data *test*, dimana perbandingan proporsi 80% untuk data *train* dan 20% untuk data *test*. Data *train* berisi 470 data gambar sedangkan untuk data *test* berisi 30 gambar.

#### 4.3 Jenis dan Sumber Data

Jenis data yang digunakan dalam penelitian ini adalah data primer. Penelitian ini menggunakan data yang diperoleh dari halaman situs *google*. Proses pengambilan data dilakukan dengan teknik *crawling* gambar.

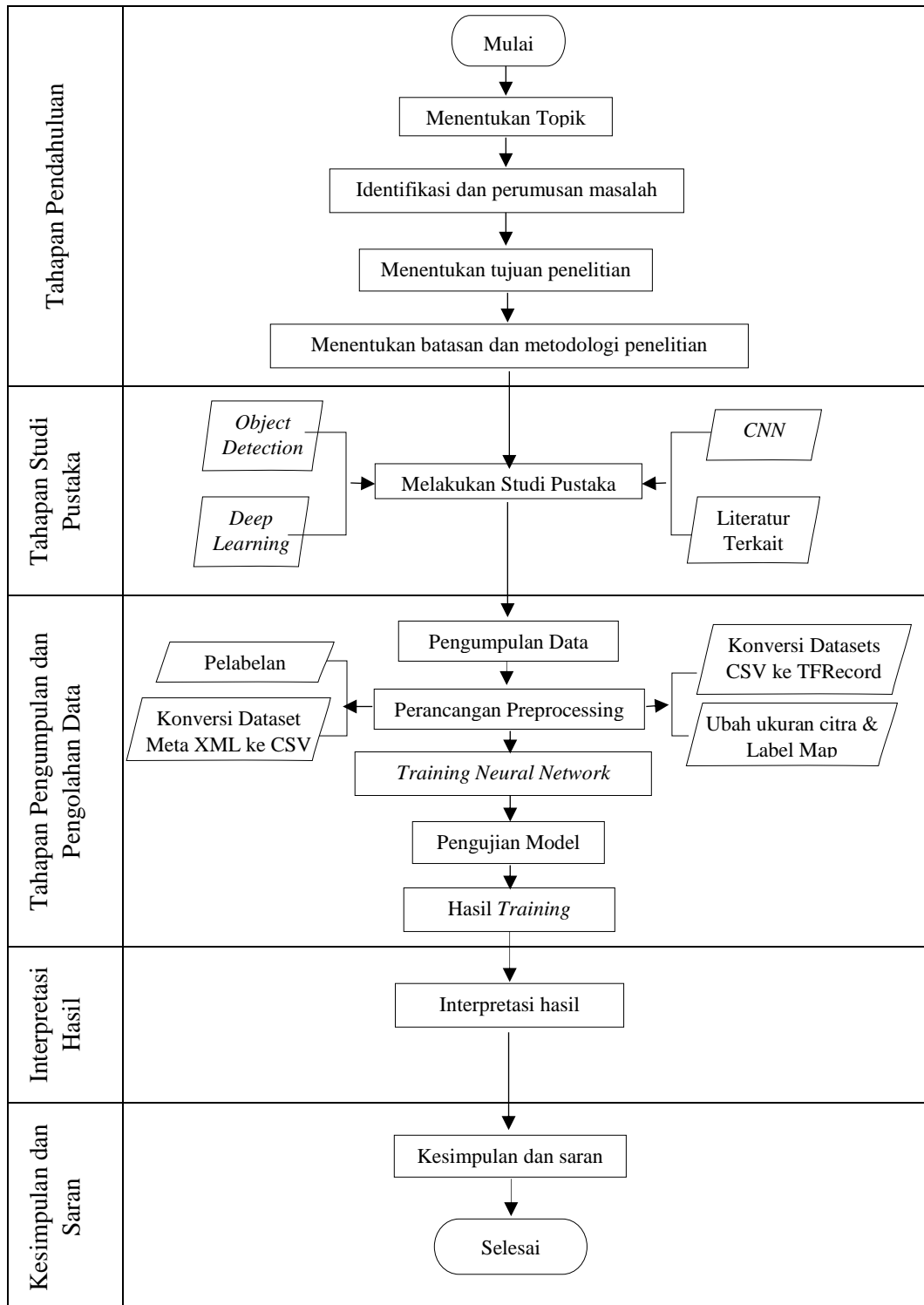
#### 4.4 Metode Analisis Data

*Software* yang digunakan dalam penelitian ini yaitu *google chrome*, *jupyter lab anaconda*, *Python 3.6.3* dan *Tensorflow 1.5.0*. Metode analisis data yang digunakan dalam penelitian ini adalah metode *Convolutional Neural Network* yang digunakan untuk mendeteksi dan mengklasifikasi objek meja dan kursi ukiran Jepara.



#### 4.5 Tahapan Penelitian

Tahapan atau langkah yang dilakukan pada penelitian ini digambarkan melalui **Gambar 4.1** berikut ini:



**Gambar 4.1** Diagram Alir Penelitian

## BAB V

### ANALISIS DAN PEMBAHASAN

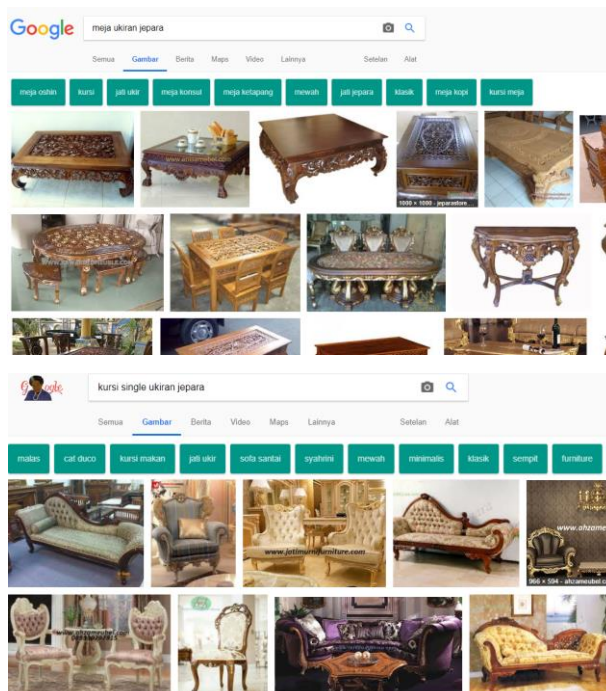
#### 5.1 Pengumpulan *Dataset*

Peneliti mengumpulkan dan membuat *dataset* berupa gambar yang di dapat dari *google image*. Peneliti menggunakan program *javascript* dan *python* untuk mengumpulkan gambar dari *google image*. Pada program *javascript* ditujukan untuk mengambil URL gambar yang ada di *google image* kemudian program *python* yang akan melakukan eksekusi untuk *download* gambar tersebut.

##### 5.1.1 Program *JavaScript*

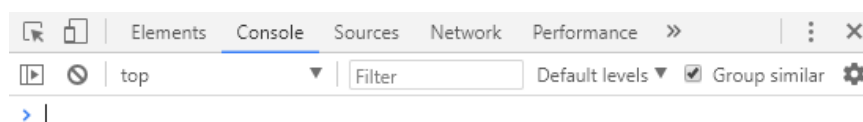
Pengambilan gambar pada *google image* dilakukan dengan menggunakan program *javascript library jQuery* untuk mendapatkan URL *image* yang akan dijadikan *dataset*. *Library jQuery* merupakan *library javascript* yang cepat, kecil, dan terdapat banyak fitur lainnya. Adapun beberapa langkah dalam mendapatkan URL *image* menggunakan *javascript library jQuery* pada *google chrome*:

1. Memasukkan *query* pada *google image* dalam hal ini yang akan dipakai adalah istilah *query* “Meja Ukiran Jepara” seperti berikut:



**Gambar 5.1** *Google Image Search*

2. Mengumpulkan URL gambar yang akan di *download* menggunakan program *python*. Setelah melakukan *search* gambar sampai batas yang ditentukan dapat membuka *developer tools – console* dengan mengetikkan *Ctrl+Shift+i* pada *keyboard*.



**Gambar 5.2** *Developer Mode Console*

3. Masukkan kode berikut pada *console google chrome*:

```

1 // pull down jquery into the JavaScript console
2 var script = document.createElement('script');
3 script.src = "https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";
4 document.getElementsByTagName('head')[0].appendChild(script);
5
6 // grab the URLs
7 var urls = $('img, .rg_di, .rg_meta').map(function() { return JSON.parse($(this).text()).ou; });
8

```

**Gambar 5.3** *Create Variable*

Dalam memasukkan kode pada *console* harus dilakukan *copy-paste* per-baris hal ini dilakukan agar tidak terjadi *error* dan URL gambar yang didapatkan akan terunduh semuanya. Peneliti membuat *variable script* dengan membuat *element* yang diberi *tagName* yaitu *script*. *tagName* merupakan *string* yang menentukan jenis elemen yang akan dibuat. Dalam sebuah dokumen HTML, *document.createElement()* menciptakan *element* HTML yang ditentukan oleh *tagName* atau HTML *unknown element* jika *tagName* yang dimasukkan tidak dikenali. Selanjutnya, *script.src* untuk menggunakan *library jQuery* dan *SP.RequestExecutor* dimana *script.src* dimasukkan kedalam *variable script*. Untuk kode baris keempat adalah cara untuk memasukkan *element* ke dalam DOM (*Document Object Model*) secara dinamis. Dalam kasus ini disisipkan di bagian akhir *<head>*, dan "*script*" adalah *string* yaitu *tagName*. Pada baris ketujuh digunakan untuk mendapatkan URL.

4. Masukkan kode terakhir pada *console google chrome* untuk mendapatkan URL gambar:

```

9 // write the URLs to file (one per line)
10 var textToSave = urls.toArray().join('\n');
11 var hiddenElement = document.createElement('a');
12 hiddenElement.href = 'data:attachment/text,' + encodeURIComponent(textToSave);
13 hiddenElement.target = '_blank';
14 hiddenElement.download = 'urls.txt';
15 hiddenElement.click();

```

**Gambar 5.4** *Download URL*

Pada baris pertama, membuat *urls* tersusun secara berurutan membentuk sebuah *array* dengan setiap URL yang ditulis pada baris baru. Pada baris kedua, data URL yang telah dibuat sebagai *array* akan diubah menjadi sebuah *hidden element* dengan *tagName a*. Pada baris ketiga digunakan untuk mengubah data URL *array* menjadi sebuah *text*. Pada baris keempat, atribut *target* digunakan untuk menentukan tempat untuk membuka dokumen seperti *\_blank* yang akan membuka *tab* baru. Kemudian pada baris kelima, digunakan untuk mengunduh dokumen dengan format *.txt*. Selanjutnya pada kode terakhir, digunakan untuk mengeksekusi atau *mendownload hiddenElement*.

### 5.1.2 Program Python

Sebelumnya, peneliti telah mempunyai file *urls* hasil *download* melalui *javascript*. Program *python* digunakan untuk *mendownload* gambar dari masing-masing *urls*. Sebelum melakukan *running* untuk *mendownload image* melalui *urls* yang didapatkan, peneliti membuat file *.py* yang berisikan kode untuk *mendownload* seluruh gambar pada *urls* yang tersedia dengan nama *download\_image.py* berikut adalah kode yang dimaksud.

```

1 from imutils import paths
2 import argparse
3 import requests
4 import cv2
5 import os
6
7 # construct the argument parse and parse the arguments
8 ap = argparse.ArgumentParser()
9 ap.add_argument("-u", "--urls", required=True, help="path to file containing image URLs")
10 ap.add_argument("-o", "--output", required=True, help="path to output directory of images")
11 args = vars(ap.parse_args())
12
13 # grab the list of URLs from the input file, then initialize the
14 # total number of images downloaded thus far
15 rows = open(args["urls"]).read().strip().split("\n")
16 total = 0

```

**Gambar 5.5** *Import Packages dan Membuat Argument*

Pada baris pertama hingga kelima merupakan *packages* atau *module* yang digunakan. *Package imutils* digunakan pada saat *list* file gambar pada folder *output*

untuk dilakukan perulangan, *request* untuk mendownload gambar yang ada di *urls*, kemudian *argsparse* untuk membuat *argument* dalam menulis *input* kode, *cv2* untuk membaca lokasi *output* gambar, *os* untuk menyimpan gambar pada sebuah folder yang ditentukan. Pada baris ketujuh hingga kesebelas, mengurai argumen baris perintah dan memuat *urls* dari *disk* ke dalam memori.

--urls merupakan *path* dari file yang berisi *urls* gambar yang dihasilkan oleh *javascript* di atas.

--output merupakan *path* dari *output* untuk menyimpan gambar yang didownload dari *google images*.

Pada baris kelima belas dan enam belas digunakan untuk memuat setiap *urls* dari file ke dalam daftar, selanjutnya menginialisasi sebuah *counter*, *total*, untuk menghitung file yang telah didownload. Pada baris ke 15, terlihat proses dalam membuka *parsing urls* kemudian membaca sebuah file tersebut (*.read()*). Selanjutnya adalah mengembalikan salinan *string* dengan karakter terdepan dan *trailing* yang dihapus (berdasarkan argument *string* yang dilewati) (*.strip()*). Kemudian menghapus “\n” pada data *urls*.

```

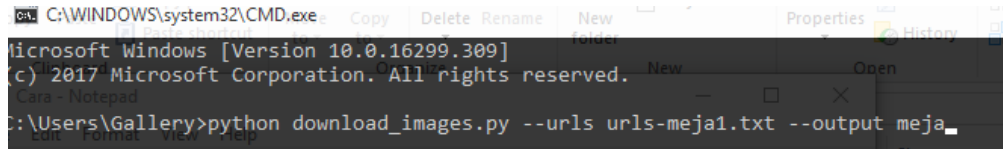
39 # loop the URLs
40 for url in rows:
41     try:
42         # try to download the image
43         r = requests.get(url, timeout=60)
44
45         # save the image to disk
46         p = os.path.sep.join([args["output"], "{}.jpg".format(
47             str(total).zfill(8))]
48
49         f = open(p, "wb")
50         f.write(r.content)
51         f.close()
52
53         # update the counter
54         print("[INFO] downloaded: {}".format(p))
55         total += 1
56
57     # handle if any exceptions are thrown during the download process
58     except:
59         print("[INFO] error downloading {...skipping".format(p))

```

**Gambar 5.6** Perulangan *Download URL*

Pada kode perulangan *download URL* digunakan untuk melakukan *download* dari sebuah URL yang telah didownload dan kemudian gambar pada URL tersebut akan disimpan pada sebuah *disk* komputer secara bertahap. Untuk melakukan *download* gambar menggunakan URL dan kode *download\_image.py* diatas dapat

mengetikkan *script* berikut pada *command prompt* untuk melakukan *download* gambar melalui *urls*.

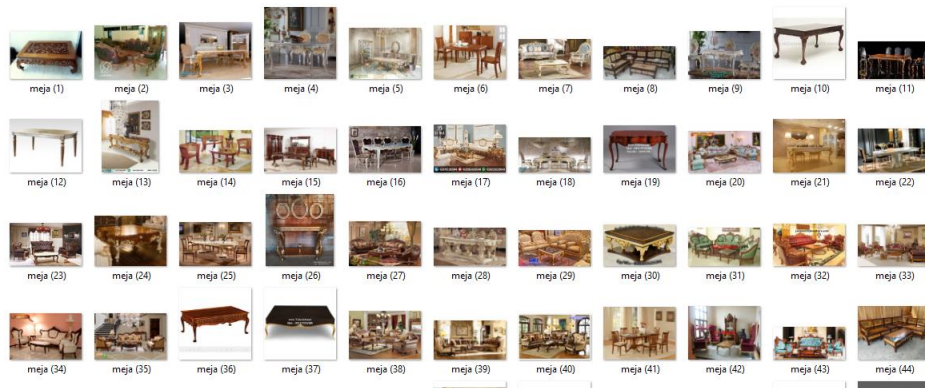


```

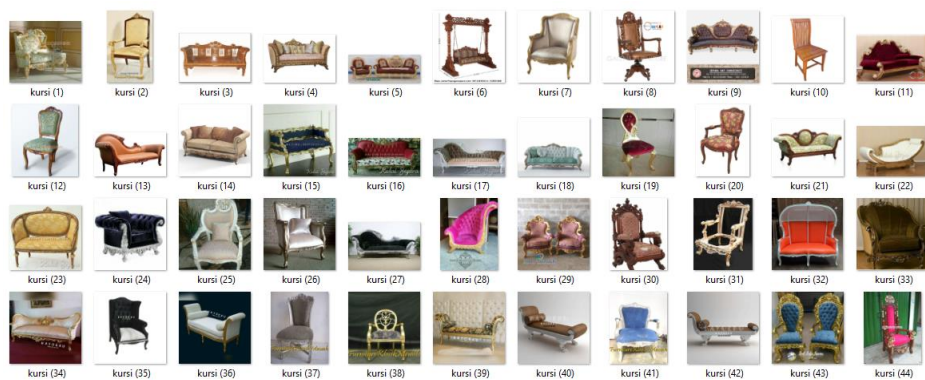
C:\WINDOWS\system32\CMD.exe Copy Delete Rename New Properties History
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.
C:\Users\Gallery>python download_images.py --urls urls-meja1.txt --output meja_
  
```

**Gambar 5.7** Perintah pada *Python*

Setelah melakukan *running script* pada *python* maka akan terbentuk sebuah *dataset* berupa citra meja dan kursi dengan motif ukiran Jepara. Berikut merupakan *dataset* peneliti yang terbentuk yaitu citra meja dan kursi dengan motif ukiran Jepara.



**Gambar 5.8** *Dataset* Citra Meja Motif Ukiran



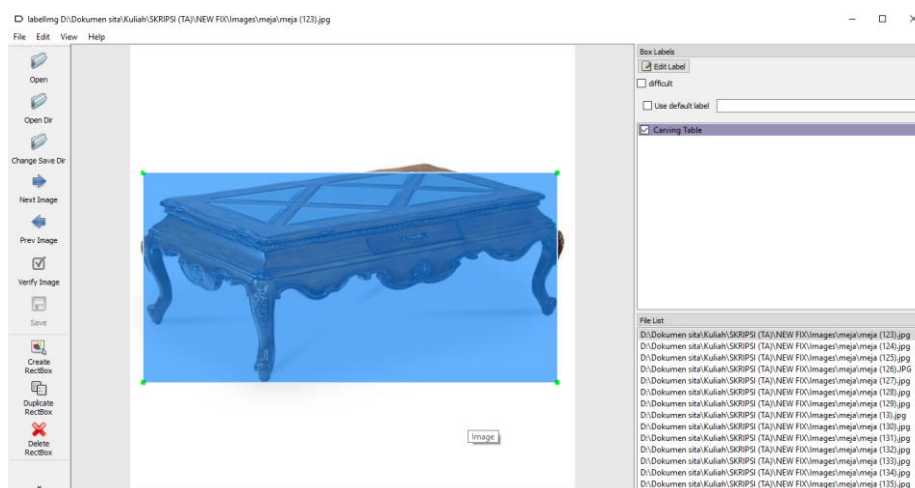
**Gambar 5.9** *Dataset* Citra Kursi Motif Ukiran

## 5.2 *Preprocessing* Citra

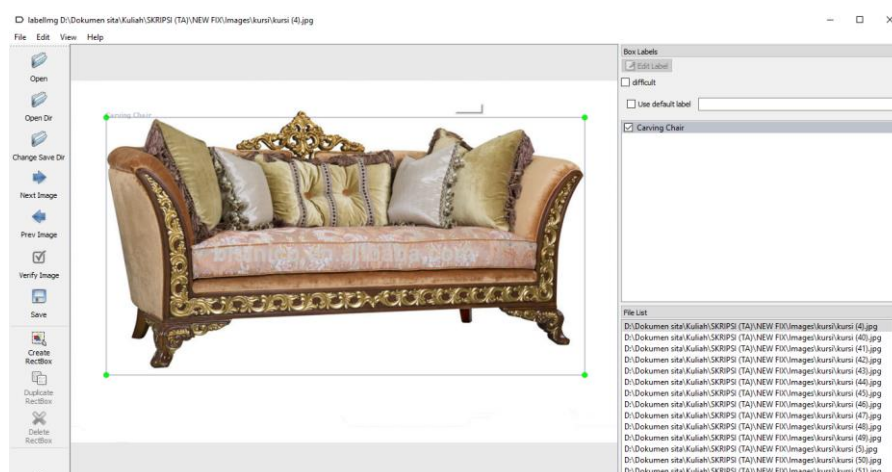
Setelah mengumpulkan *dataset*, selanjutnya dilakukan *preprocessing* citra dengan beberapa tahapan seperti berikut:

### 5.2.1 Pelabelan Gambar/Citra

Pelabelan gambar adalah tahap awal dimana *dataset input* diberikan label atau pengenal (tanda) dengan tujuan untuk menyimpan informasi gambar yang selanjutnya disimpan dalam berkas XML dengan format *PASCAL VOC*. Pelabelan dilakukan secara manual terhadap 500 dataset citra meja dan kursi dengan motif ukiran Jepara menggunakan *labelImg*.



Gambar 5.10 Proses Pelabelan Gambar Meja



Gambar 5.11 Proses Pelabelan Gambar Kursi

### 5.2.2 Konversi *Dataset Meta XML* ke CSV

Setelah dilakukan pelabelan perlu adanya konversi berkas dari XML ke CSV untuk tujuan konversi *dataset* ke berkas *TfRecord*, berikut kode untuk mengkonversi berkas *train* dan *test* XML ke CSV.

```

1 | import os
2 | import glob
3 | import pandas as pd
4 | import xml.etree.ElementTree as ET
5 |
6 | def xml_to_csv(path):
7 |     xml_list = []
8 |     for xml_file in glob.glob(path + '/*.xml'):
9 |         tree = ET.parse(xml_file)
10 |         root = tree.getroot()
11 |         for member in root.findall('object'):
12 |             value = (root.find('filename').text,
13 |                    int(root.find('size')[0].text),
14 |                    int(root.find('size')[1].text),
15 |                    member[0].text,
16 |                    int(member[4][0].text),
17 |                    int(member[4][1].text),
18 |                    int(member[4][2].text),
19 |                    int(member[4][3].text)
20 |             )
21 |             xml_list.append(value)
22 |     column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
23 |     xml_df = pd.DataFrame(xml_list, columns=column_name)
24 |     return xml_df
25 |
26 | def main():
27 |     for directory in ['train', 'test']:
28 |         image_path = os.path.join(os.getcwd(), 'annotations/{}'.format(directory))
29 |         xml_df = xml_to_csv(image_path)
30 |         xml_df.to_csv('data/{}_labels.csv'.format(directory), index=None)
31 |         print('Successfully converted xml to csv.')
32 |
33 | main()

```

**Gambar 5.12** Kode Konversi XML ke CSV

### 5.2.3 Konversi Dataset CSV ke TFRecord

Setelah proses konversi berkas XML dengan *output* berupa file CSV perlu adanya konversi ke *TensorFlow Record* file yang digunakan untuk *feeding* data pada proses *training*, berikut adalah kode pembuatan *TFRecord*:

```

80 | def main():
81 |     writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
82 |     path = os.path.join(os.getcwd(), 'images/{}'.format(FLAGS.type))
83 |     examples = pd.read_csv(FLAGS.csv_input)
84 |     grouped = split(examples, 'filename')
85 |     for group in grouped:
86 |         tf_example = create_tf_example(group, path)
87 |         writer.write(tf_example.SerializeToString())
88 |
89 |     writer.close()
90 |     output_path = os.path.join(os.getcwd(), FLAGS.output_path)
91 |     print('Successfully created the TFRecords: {}'.format(output_path))
92 |
93 | if __name__ == '__main__':
94 |     tf.app.run()

```

**Gambar 5.13** Kode Konversi CSV ke TFRecord

### 5.2.4 Label Map

*Dataset* (file *TFRecord*) dan peta labelnya yang sesuai dengan proses pelabelan gambar. Berikut ini adalah peta label yang digunakan dan peta label memiliki dua kelas yaitu untuk meja dinamakan dengan “*Carving Table*” sedangkan untuk kursi dinamakan dengan “*Carving Chair*” yang disimpan pada berkas dengan format “.pbtxt” yang selanjutnya dibutuhkan pada saat konfigurasi *pipeline*.



```

1 item {
2   id: 1
3   name: 'Carving Chair'
4 }
5
6 item {
7   id: 2
8   name: 'Carving Table'
9 }

```

**Gambar 5.14** Kode Konfigurasi *Label Map*

## 5.3 Pengolahan Citra

### 5.3.1 Konfigurasi *Object Detection Training Pipeline*

API Deteksi Objek *Tensorflow* menggunakan berkas *protobuf* untuk mengkonfigurasi proses pelatihan dan evaluasi. Pada tingkat tinggi, file konfigurasi dibagi menjadi 5 bagian yaitu:

1. *Model*: mendefinisikan jenis model apa yang akan dilatih (yaitu *meta-architecture, feature extractor*)
2. *Training\_config*: menentukan parameter apa yang harus digunakan untuk melatih parameter model (misalnya parameter SGD, *input preprocessing* dan nilai inisialisasi *extractor* fitur).
3. *Eval\_config*: menentukan metrik pengukuran apa yang akan dilaporkan untuk evaluasi.
4. *Train\_input\_config*: mendefinisikan *dataset* apa yang harus dilatih modelnya.
5. *Eval\_input\_config*: mendefinisikan *dataset* apa yang akan dievaluasi model. Biasanya ini harus berbeda dari *dataset* masukan pada *training*.

```

feature_extractor {
  type: 'ssd_mobilenet_v1'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
      l2_regularizer {
        weight: 0.00004
      }
    }
    initializer {
      truncated_normal_initializer {
        stddev: 0.03
        mean: 0.0
      }
    }
    batch_norm {
      train: true,
      scale: true,
      center: true,
      decay: 0.9997,
      epsilon: 0.001,
    }
  }
}

```

**Gambar 5.15** Kode Konfigurasi *Pipeline*

### 5.3.2 Training Neural Network

Tahap pelatihan *Neural Network* adalah tahap utama dimana sebuah *Neural Network* dilatih untuk mempelajari suatu pola yang diharapkan menghasilkan suatu pengenalan deteksi objek yang sesuai dengan yang diharapkan dengan tingkat akurasi yang tinggi.

```
def main():
    assert FLAGS.train_dir, '`train_dir` is missing.'
    if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
    if FLAGS.pipeline_config_path:
        configs = config_util.get_configs_from_pipeline_file(
            FLAGS.pipeline_config_path)
        if FLAGS.task == 0:
            tf.gfile.Copy(FLAGS.pipeline_config_path,
                os.path.join(FLAGS.train_dir, 'pipeline.config'),
                overwrite=True)
    else:
        configs = config_util.get_configs_from_multiple_files(
            model_config_path=FLAGS.model_config_path,
            train_config_path=FLAGS.train_config_path,
            train_input_config_path=FLAGS.input_config_path)
        if FLAGS.task == 0:
            for name, config in [('model.config', FLAGS.model_config_path),
                ('train.config', FLAGS.train_config_path),
                ('input.config', FLAGS.input_config_path)]:
                tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                    overwrite=True)
```

Gambar 5.16 Kode Program Proses Training

### 5.3.3 Export Graph Model

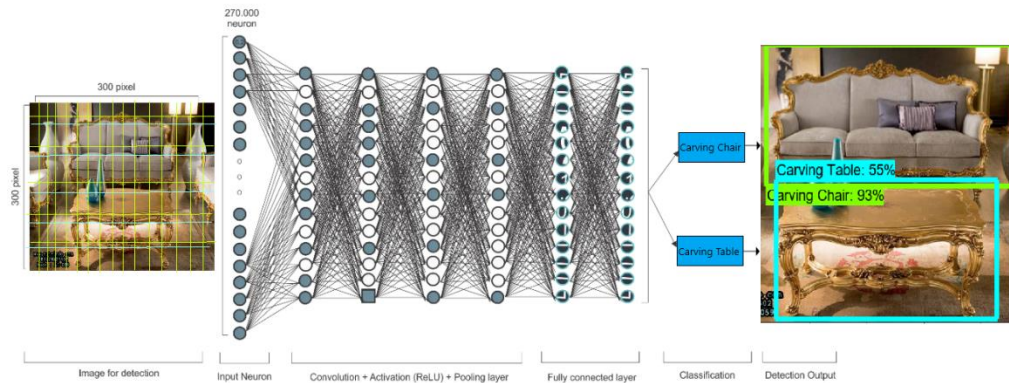
Pada saat proses *training* maka akan menghasilkan *checkpoint* yang dibuat secara otomatis oleh *Tensorflow* berbentuk *graph tensor* yang bertujuan untuk menyimpan informasi proses *training* yang dilakukan, jika proses *training* selesai maka selanjutnya adalah mengekspor *graph tensor* dan dijadikan model yang siap dipakai.

```
def main():
    pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
    with tf.gfile.GFile(FLAGS.pipeline_config_path, 'r') as f:
        text_format.Merge(f.read(), pipeline_config)
    if FLAGS.input_shape:
        input_shape = [
            int(dim) if dim != '-1' else None
            for dim in FLAGS.input_shape.split(',')
        ]
    else:
        input_shape = None
    exporter.export_inference_graph(FLAGS.input_type, pipeline_config,
        FLAGS.trained_checkpoint_prefix,
        FLAGS.output_directory, input_shape)

if __name__ == '__main__':
    tf.app.run()
```

Gambar 5.17 Kode Program Export Graph Model

### 5.3.4 Arsitektur Jaringan

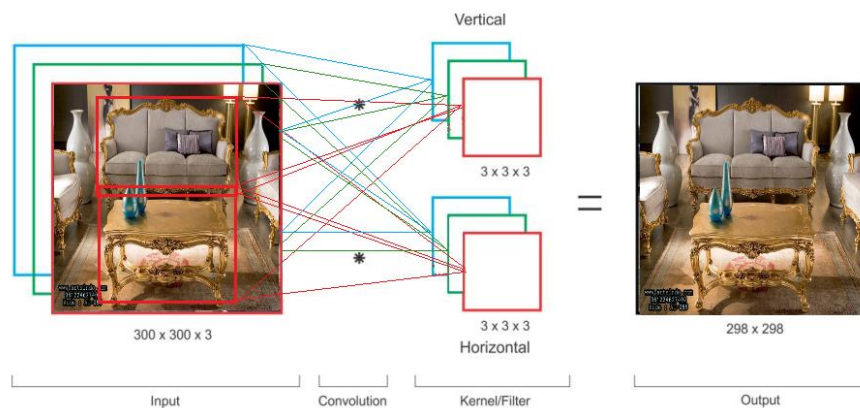


**Gambar 5.18** Arsitektur Jaringan

Pada Gambar 5.18 merupakan arsitektur jaringan dari *convolutional neural network* yang memiliki beberapa bagian yaitu *image for detection*, *input neuron*, *convolution + activation (ReLU) + pooling layer*, *fully connected layer*, *classification* dan *detection output*.

Pada bagian *image for detection* yaitu gambar/citra yang akan di deteksi dilakukan *resize* sebesar 300x300 *pixels* dengan warna RGB (*Red, Green, Blue*) dengan *channel* sebanyak 3 sehingga yang masuk ke dalam *layer* pertama atau bagian *input neuron* sebanyak 270.000 *neuron* yang diperoleh dari perhitungan 300x300x3. Setiap *neuron* memiliki nilai parameter dimana parameter yang digunakan dalam jaringan tersebut berkisar antara 0 sampai 1. Proses *resizing* bertujuan untuk menyamakan ukuran citra.

#### 5.3.4.1 Convolution

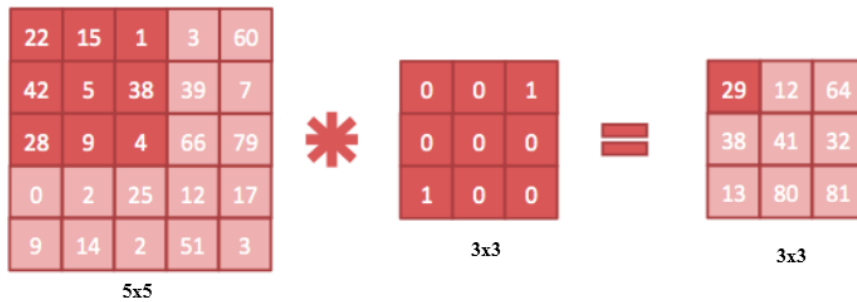


**Gambar 5.19** Convolutional Layer

*Convolution layer* terdiri dari *neuron* yang tersusun sehingga membentuk sebuah *kernel (filter)* dengan panjang dan tinggi (*pixels*). *Convolution layer* melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi (*Convolution*) didefinisikan sebagai cara untuk mengkombinasikan dua buah deret angka yang menghasilkan deret angka yang ketiga. Dalam penelitian ini, dua buah deret angka tersebut terdapat dalam *input* dan *kernel (filter)* sedangkan deret angka yang ketiganya terdapat di *output*. *Input* dan *kernel (filter)* keduanya memiliki deret angka berupa matriks. Pada *input*, deret angka yang diperoleh berdasarkan tingkat warna yang ada pada masing-masing *pixels* sedangkan pada *kernel (filter)* deret angka disesuaikan oleh kebutuhan peneliti. Terdapat beberapa jenis *kernel (filter)* yang biasa digunakan yaitu operasi *identity*, *edge detection*, *sharpen box blur*, *Gaussian blur*, dan lain sebagainya.

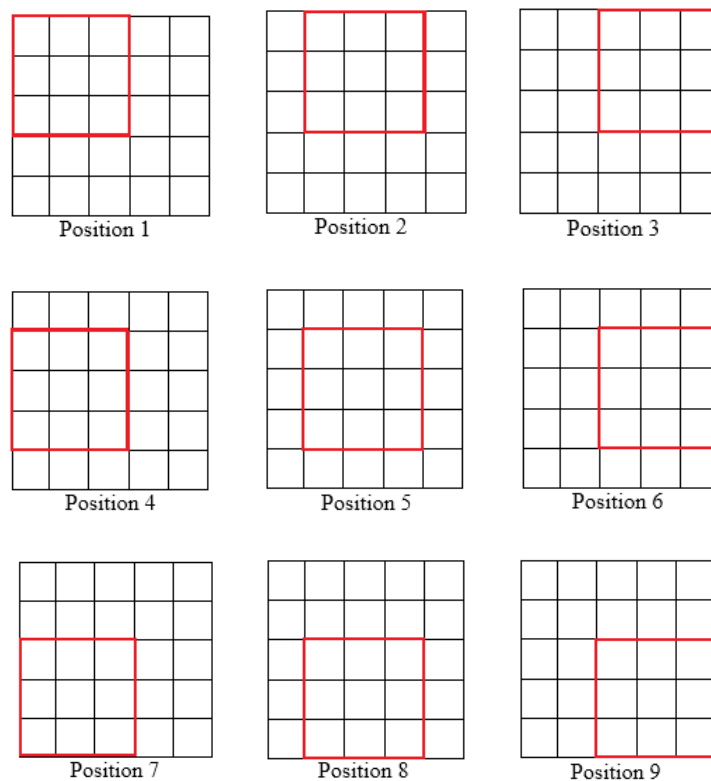
Pada Gambar 5.19, pada bagian *input*, gambar/citra memiliki tinggi 300 dan lebar 300 dengan kedalaman sebesar 3 *channel* yaitu RGB atau merah, hijau dan biru. Lapisan konvolusi dibentuk dengan menjalankan *filter* di atasnya. *Filter* merupakan blok lain atau kubus dengan tinggi dan lebar yang lebih kecil namun memiliki kedalaman yang sama di atas gambar asli atau dasar. *Filter* digunakan untuk menentukan pola apa yang akan dideteksi yang selanjutnya dilakukan konvolusi kembali atau dikalikan dengan nilai matriks *input*. Nilai pada masing-masing kolom dan baris pada matriks sangat bergantung pada jenis pola yang akan dideteksi.

Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input*. Konvolusi akan menghasilkan transformasi *linear* dari data *input* sesuai informasi spasial pada data. Bobot pada *layer* tersebut menspesifikasikan *kernel* konvolusi yang digunakan, sehingga *kernel* konvolusi dapat dilatih berdasarkan *input* pada CNN.



**Gambar 5.20** Proses Konvolusi

Pada Gambar 5.20 merupakan contoh proses konvolusi untuk dapat lebih memahami cara kerja proses konvolusi. Peneliti menggunakan sampel deret angka pada *input* dengan ukuran 5x5 dikarenakan keterbatasan penulisan dengan ukuran 300x300 dan menggunakan *kernel (filter)* dengan ukuran 3x3 untuk mendeteksi garis lurus keatas (*vertical edge detection*). Dengan menggunakan *kernel (filter)* ukuran 3x3, *strided* atau langkah yang digunakan dalam perhitungan konvolusi tersebut adalah 1 maka proses perhitungan konvolusi tersebut dapat divualisasikan seperti gambar berikut:



**Gambar 5.21** Posisi Proses Konvolusi

Pada proses konvolusi, *kernel (filter)* berukuran 3x3 bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada Gambar 5.20 di sebelah kanan. Pada Gambar 5.20, *filter* yang digunakan peneliti tidak lebih dari satu set bobot/*weight* yaitu (3x3x3)+1 bias = 10 bobot/*weight*. Pada masing-masing posisi, jumlah piksel yang dihitung menggunakan rumus  $\sum_i^n = W_i X_i + b$  dan kemudian nilai yang baru akan diperoleh. Ukuran gambar yang dihasilkan dari proses konvolusi semakin menyusut secara terus menerus atau berurutan, hal ini tidak begitu baik karena ukurannya akan menjadi semakin kecil. Selain itu akan membatasi penggunaan *filter* ukuran besar karena akan menghasilkan pengurangan ukuran lebih cepat. Untuk mencegah permasalahan ini, pada umumnya peneliti hanya menggunakan langkah 1.

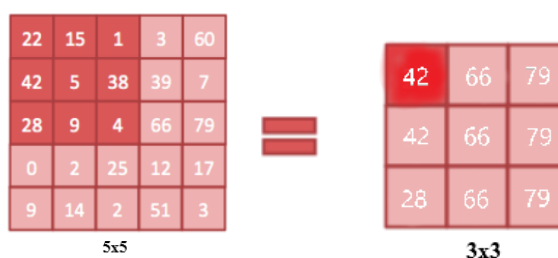
#### 5.3.4.2 Activation Function

Fungsi aktivasi memiliki tugas untuk membuat *neural network* menjadi *non-linear* pada nilai hasil konvolusi. Pada penelitian ini, peneliti menggunakan fungsi aktivasi reLU. ReLU sendiri memiliki rumus sebagai berikut:

$$f(x) = \max(x, 0)$$

Dimana  $x$  merupakan *input neuron*. Angka 0 pada rumus reLU merupakan unit linier yang dikoreksi jika *input* kurang dari 0. Artinya, jika *input* lebih besar dari 0, *outputnya* sama dengan *input*. Aktivasi reLU pada dasarnya merupakan sebuah fungsi aktivasi *non-linear* yang paling sederhana. Bila mendapat *input* positif, turunannya hanya 1, dengan kata lain, aktivasi hanya men-*threshold* pada nilai nol. Penelitian menunjukkan bahwa reLU menghasilkan pelatihan yang lebih cepat untuk jaringan besar.

#### 5.3.4.3 Pooling Layer

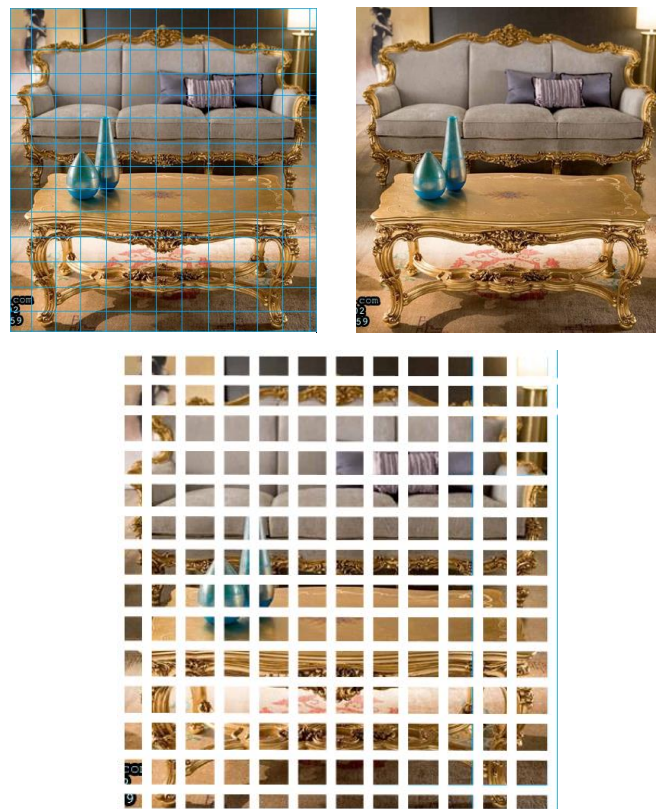


**Gambar 5.22** Pooling Layer

Pada tahap *pooling layer* digunakan untuk mengurangi gambar. Pada Gambar 5.22 merupakan *pooling layer* menggunakan metode *max pooling* yang bertujuan untuk menentukan nilai maksimum pada setiap nilai *input* yang telah dikonvolusi dengan *filter*. Pada Gambar 5.22 terdapat lapisan dengan ukuran 5x5 apabila peneliti menggunakan *filter* 3x3 dengan *stride* 1 maka diperoleh hasil *max pooling* dengan ukuran 3x3.

#### 5.3.4.4 Fully Connected Layer

Pada tahap akhir *convolution layer* dan *pooling layer*, jaringan umumnya menggunakan lapisan yang terhubung sepenuhnya di mana setiap piksel dianggap sebagai *neuron* terpisah seperti jaringan saraf biasa. Lapisan terakhir sepenuhnya akan mengandung banyak *neuron* sebagai jumlah kelas yang harus diprediksi. Pada proses *fully connected layer*, piksel yang dianggap sebagai meja dan kursi yang terpisah akan disatukan menjadi sebuah *output* yang terdiri dari dua label kelas yaitu *carving chair* dan *carving table* sehingga lapisan yang terhubung memiliki 2 *neuron output*.



Gambar 5.23 Fully Connected Layer

### 5.3.4.5 Classification

Tahapan klasifikasi akan menentukan bagian mana saja pada setiap piksel yang memiliki pola meja dan kursi. Dalam penelitian ini peneliti menguji gambar dengan masing-masing gambar meja dan kursi dengan gambar motif ukiran Jepara selain itu peneliti juga menguji gambar dengan sebuah gambar yang terdiri dari 2 klasifikasi secara langsung yaitu meja dan kursi dengan gambar motif ukiran Jepara untuk menentukan apakah gambar yang diujikan termasuk klasifikasi meja atau kursi atau keduanya.

### 5.3.4.6 Detection Output

*Detection Output* merupakan hasil akhir / tahap akhir dari deteksi meja dan kursi motif ukiran Jepara. Untuk melihat dan menguji coba hasil model yang di *training* maka diperlukan sebuah program untuk memuat dan menjalankan model (*frozen inference graph*) tersebut dan menampilkan hasil gambar yang telah diberikan hasil deteksi berupa kotak hijau dengan nilai persentasenya.

## 5.4 Model Hasil Training

Pembahasan model merupakan hasil implementasi dan proses pelatihan. Berikut ini akan dijabarkan mengenai model yang telah diperoleh peneliti:

### 5.4.1 Training Steps

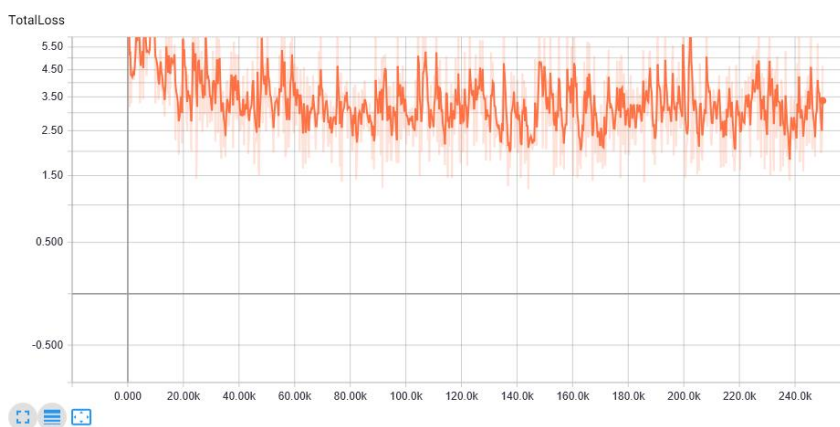


Gambar 5.24 Grafik Global Training Step



Pada Gambar 5.24 merupakan grafik dari *training steps* yang dapat dilihat melalui *tensorboard*. *Training steps* merupakan langkah dimana proses *training* dilakukan yang memvisualisasikan hasil *training*.

### 5.4.2 Total Loss



**Gambar 5.25** Grafik *Total Loss*

Pada Gambar 5.25 merupakan grafik *total loss* yang dihasilkan pada saat melakukan proses *training* sampai dengan selesai sesuai dengan jumlah iterasi yang dilakukan yaitu sebanyak 250.000 langkah/*steps*. Dapat disimpulkan bahwa rata-rata nilai *loss* dari *step* pertama hingga *step* terakhir adalah 3.50 hingga 1.55.

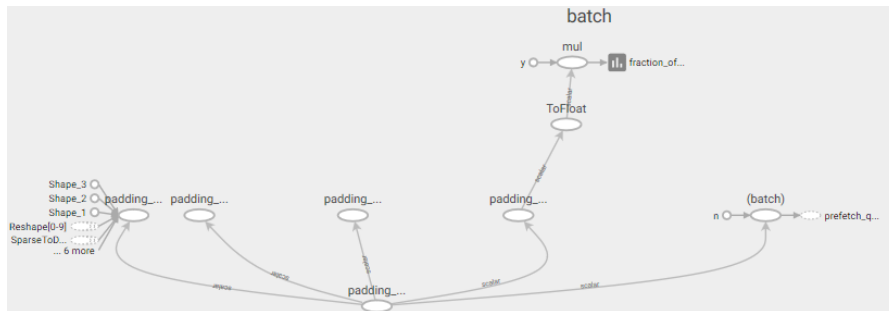
### 5.4.3 Tensor Graph

Proses *training* dari sebuah pembelajaran *Neural Network* menggunakan *Tensorflow* dapat menghasilkan *graph tensor* yang merepresentasikan langkah atau alat yang dilakukan pada proses *training*, berikut ini adalah daftar *legend* pada *graph tensor*:

Symbol	Meaning
	High-level node representing a name scope. Double-click to expand a high-level node.
	Sequence of numbered nodes that are not connected to each other.
	Sequence of numbered nodes that are connected to each other.
	An individual operation node.
	A constant.
	A summary node.
	Edge showing the data flow between operations.
	Edge showing the control dependency between operations.
	A reference edge showing that the outgoing operation node can mutate the incoming tensor.

**Gambar 5.26** *Graph Legend*

### 5.4.3.1 Batch



**Gambar 5.27** Batch Graph

Pada Gambar 5.27 terdapat 5 *padding* sebagai sebuah *node* operasi individu dimana setiap *padding* menunjukkan arus data antar operasi. Beberapa *padding* tersebut mengalirkan *scalar* ke data pengoperasian lain dengan keterangan sebagai berikut:

- Padding* pertama mengalirkan data untuk operasi berupa *scalar* ke 4 *padding* lainnya dan mengalirkan *scalar* ke sebuah *batch* dimana *batch* tersebut dialirkan data dengan nilai konstan sebesar nol dan *batch* mengalirkan data operasi ke *prefetch\_q*.
- Padding* kedua mengalirkan data untuk operasi berupa *scalar* ke *shape*, *shape 5*, *shape 6*, and *reshape 0...*, *sparseTo* dan 11 operasi lainnya.
- Padding* ketiga mengalirkan data untuk operasi berupa *scalar* ke operasi lainnya hanya mengoperasikan data pada *padding* itu sendiri.
- Padding* kelima mengalirkan data *scalar* ke *ToFloat* dan kemudian *ToFloat* mengalirkan data *scalar* ke *mul*, dimana *mul* tersebut dialirkan data dengan nilai *y* konstan sebesar nol dan kemudian *mul* mengalirkan data operasi berupa sebuah ringkasan atau *fraction\_of*.

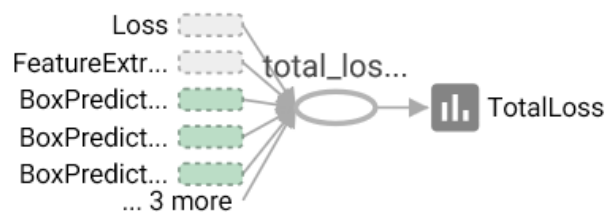
### 5.4.3.2 Train Step



**Gambar 5.28** Train Step Graph

Pada Gambar 5.28 merupakan tahapan *training*, dimana terdapat 3 *node* operasi individu yaitu *FloorMod*, *Equal* dan *GreaterE*. *FloorMod* dialiri dengan nilai *y* konstan sebesar nol dan dialiri dengan *node* tingkat tinggi yaitu *global\_step* kemudian *FloorMod* mengalirkan data *scalar* ke *Equal* dengan dialiri nilai konstan *y* sebesar nol. *GreaterE* dialiri dengan nilai *y* konstan sebesar nol dan dialiri dengan *node* tingkat tinggi yaitu *global\_step*.

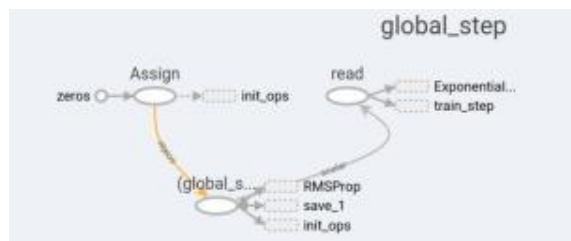
#### 5.4.3.3 Total Loss



**Gambar 5.29** Total Loss Graph

Pada Gambar 5.29 menunjukkan bahwa terdapat 6 *BoxPredict*, *Loss* dan *Feature* yang mengalirkan data ke dalam operasi *total\_loss* yang kemudian *total\_loss* menghasilkan *node* ringkasan *total\_loss*.

#### 5.4.3.4 Global Step



**Gambar 5.30** Global Step Graph

Pada grafik *global step graph* diatas menunjukkan *global step* atau langkah yang dilakukan secara global atau umum. Dalam grafik tersebut tersebut terdapat 3 operasi yaitu *global\_step*, *assign* dan *read*. Operasi *global\_step* mengalirkan *node* tingkat tinggi yang mewakili cakupan nilai yaitu pada *RMSProp*, *save\_1*, *init\_ops*. Kemudian *global\_step* mengalirkan nilai *scalar* ke operasi *read* dengan *exponential* dan *train\_step*. Operasi *assign* berasal dari *global\_step* dengan ujung referensi yang menunjukkan bahwa *node* operasi keluar mengubah tensor yang masuk kemudian *assign* dialiri *zeros* dengan nilai konstan sebesar nol dan *assign* menunjukkan ketergantungan control antar operasi dengan *init\_ops*.

#### 5.4.4 Model

Hasil akhir dari proses pembelajaran atau pelatihan *Neural Network* adalah terbentuknya sebuah model yang siap pakai untuk pendeteksian lebih lanjut atau dengan kata lain disebut dengan *testing*. Model yang dimaksudkan pada *Tensorflow* API adalah berupa *file checkpoint* hasil *training*/pelatihan dan data *tensor graph* yang dimuat pada berkas berekstensi *protobuf* “.pb”.

```

model
├── checkpoint
├── frozen_inference_graph.pb
├── model.ckpt.data-00000-of-00001
├── model.ckpt.index
├── model.ckpt.meta
└── saved_model
    ├── saved_model.pb
    └── variables

2 directories, 6 files

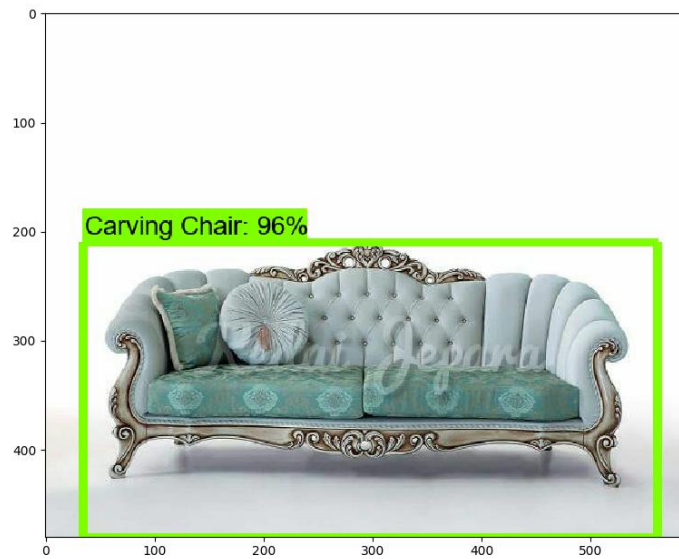
```

**Gambar 5.31** Direktori Model Hasil *Training*

#### 5.5 Hasil Deteksi

Hasil uji coba dari sebuah model yang telah dilakukan pelatihan atau *training* akan siap dipakai untuk dilakukan pendeteksian atau *testing* terhadap keberadaan sebuah meja dan kursi pada suatu *frame* gambar dengan ditandainya sebuah kotak warna hijau beserta dengan persentase akurasinya.

Nilai akurasi pada hasil deteksi didapatkan dari sebuah *class* yang bernama *graph()*, *class* tersebut bertugas untuk mengkomputasi nilai keluaran pada *Neural Network* yang merepresentasikan *dataflow* berupa *graph*, *graph* yang dimaksud adalah *graph* yang sudah di *training* sebelumnya yang berupa *checkpoint* pada saat proses *training* kemudian diekspor ke *graph inference*. Setelah komputasi tersebut selesai, *class* ini akan memanggil *tensor* berdasarkan nama yang mengembalikan data berupa nama *tensor* yaitu “*detection\_scores:0*”, skor diinisialisasikan dengan angka 0 agar persentase hasil yang dikembalikan dimulai dari angka 0% hingga 100%. Berikut merupakan hasil uji coba model untuk mendeteksi meja dan kursi motif ukiran Jepara pada sebuah *frame* gambar.

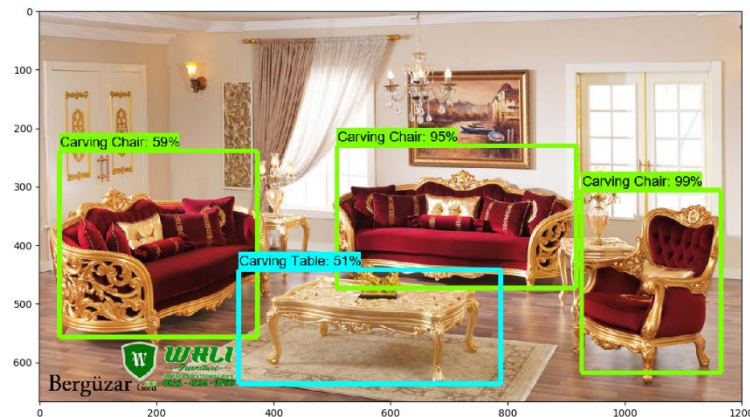


**Gambar 5.32** Hasil Deteksi Kursi Motif Ukiran Jepara



**Gambar 5.33** Hasil Deteksi Meja Motif Ukiran Jepara

Berdasarkan Gambar 5.32 dan Gambar 5.33, diperoleh hasil akurasi sebesar 96% pada citra kursi motif ukiran Jepara (*Carving Chair*) dan 99% pada citra meja motif ukiran Jepara (*Carving Table*). Berikut ini merupakan hasil deteksi pada sebuah citra/gambar yang memiliki 2 objek baik itu kursi maupun meja pada satu *frame* gambar. Hasil deteksi pada kedua gambar tersebut menghasilkan nilai akurasi yang cukup tinggi.



**Gambar 5.34** Hasil Deteksi Meja dan Kursi Motif Ukiran Jepara

Berdasarkan Gambar 5.34 diperoleh hasil akurasi sebesar 51% pada meja (*Carving Table*) dan 59%-99% pada kursi (*Carving Chair*). Hasil deteksi pada *frame* gambar diatas dengan 2 objek meja dan kursi menghasilkan akurasi yang berbeda-beda, hal ini dikarenakan kurangnya jumlah *training* yang diberikan/dilakukan. Berikut ini adalah hasil deteksi meja dan kursi motif ukiran Jepara menggunakan video:



**Gambar 5.35** Hasil Deteksi Menggunakan Video

Berdasarkan pada Gambar 5.35 diperoleh hasil 80% pada kursi dan 79% pada meja. Hasil deteksi meja dan kursi pada *frame* gambar maupun video menghasilkan nilai akurasi yang cukup tinggi. Hasil deteksi yang didapatkan pada video memberikan hasil akurasi yang berbeda-beda pada saat objek yang dideteksi berpindah posisi dikarenakan kurangnya jumlah *steps* saat melakukan *training* dan pengambilan posisi objek yang dideteksi berpindah-pindah.

## BAB VI

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Berdasarkan hasil analisis yang telah dilakukan, maka dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Model yang terbentuk adalah model hasil *training* dengan jumlah 250000 *steps* dengan 2 *batch size* yaitu berupa *graph inference* yang terdiri dari *file checkpoint*, *frozen\_inference\_graph.pb*, dan terdapat 3 file *model-ckpt* yang masing-masing berekstensi *.data-00000-of-00001*, *.index*, dan *.meta* yang diletakkan dalam satu direktori yang sama dan digunakan untuk melakukan pengujian model pada saat melakukan pendeteksian objek.
2. Hasil dari pendeteksian klasifikasi meja dan kursi pada suatu citra digital menggunakan *Convolutional Neural Network* dapat dinilai bekerja dengan baik.
3. Tingkat akurasi model yang didapatkan dari hasil pendeteksian klasifikasi citra meja dan kursi motif ukiran Jepara pada suatu citra digital menggunakan *Convolutional Neural Network* berkisar antara 70% hingga 99%.

#### 6.2 Saran

Berdasarkan penelitian yang telah dilakukan, maka dapat diberikan beberapa saran sebagai berikut:

1. Menambahkan jumlah *dataset* dan ragam objek pada gambar untuk melatih model dan mencapai akurasi yang tinggi.
2. Menambahkan jumlah pada *step training* sehingga menghasilkan hasil akurasi yang lebih tinggi dengan menggunakan jumlah *batch* hingga 64/128.
3. Mengembangkan kembali pengenalan objek dengan fokus untuk mendeteksi atau mengenali motif ukiran, baik itu motif ukiran asal Jepara atau motif ukiran lainnya yang terdapat pada berbagai macam *furniture*.
4. Menggunakan spesifikasi perangkat yang lebih tinggi yaitu dengan menggunakan komputer dengan *Random Access Memory (RAM)* yang tinggi dan menggunakan *Graphics Processing Unit (GPU)* untuk mempercepat proses *training*.

## DAFTAR PUSTAKA

- Abhirawa, Halprin., et.al. *Pengenalan Wajah Menggunakan Convolutional Neural Network*. E-Proceeding of Engineering: Vol.4, No.3 Desember 2017 ISSN 2355-9365
- Adianshar, Ammar. 2014. *Penerapan Recurrent Neural Network Dalam Identifikasi Tulisan Tangan Huruf Jepang Jenis Katakana*. Skripsi. Program Studi Teknologi Informasi Fakultas Ilmu dan Teknologi Informasi Universitas Sumatera Utara.
- Aryanto, Yunus. *173 meja & kursi*. Griya Kreasi (Penebar Swadaya Grup).
- Azrorry, Muhammad. 2015. *Motif Ukir Jepara*. Diakses pada 15 April 2018 dari <http://bloggazrorry.blogspot.co.id/2015/04/motif-ukir-jepara.html>
- Danukusumo, Kefin Pudi. 2017. *Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Citra Candi Berbasis GPU*. Tugas Akhir. Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta.
- Deng, L. & Yu, D., 2014. *Deep Learning: Methods and Application, Foundations and Trends in Signal Processing*.
- Dobrev. 2004. *Artificial Intelligence*. Binus University.
- Dosovitskiy, Alexey., et.al. *Learning to Generate Chairs, Tables and Cars with Convolutional Networks*. IEEE Transactions on Pattern Anlysis and Machine Intelligence.
- Fajri, Nazar Iskandar. 2011. *Prediksi Suhu dengan Menggunakan Algoritma-Algoritma yang Terdapat pada Artificial Neural Network*. Thesis. Bandung, Indonesia: Institut Teknologi Bandung.
- Goodfellow, dkk. 2016. *Deep Learning*. MIT Press.
- Habibi, Anjar Ahmad. 2016. *Jepara Carving Centre*. Tesis. Program Studi Arsitektur Fakultas Teknik Universitas Muhammadiyah Surakarta.
- Haykin, S. 2009. *Neural Networks and Learning Machines*. United State of America: Pearson.
- Hermawan, Arief. 2006. *Jaringan Saraf Tiruan Teori dan Aplikasi*. Yogyakarta: Penerbit ANDI.



- Jalled, Fares. 2016. *Object Detection Using Image Processing*. Diakses dari <https://arxiv.org/pdf/1611.07791.pdf>
- Kristiyani, Devina. 2014. *Perancangan Kampanye Pengenalan Seni Ukir Jepara Kepada Anak Usia 9-12 Tahun di Kota Jepara*. Skripsi. Fakultas Seni Universitas Kristen Maranatha.
- Ldya, et al. 2010. *Pengertian Citra*. Universitas Sumatera Utara.
- LeCun et al. 2015. *Deep Learning*. Nature, 521 (7533) 436-444.
- Marques, O., & Borivoje Furht. 2002. *Content-Based Imaged and Video Removal*. Florida Atlantic University Boca Raton, FL, USA : Kluwer Academic Publisher.
- McCarthy. 2007. *Artificial Intelligence*. Binus University.
- Mitbal. 2014. *Apa itu Machine Learning*. Diakses pada 8 Maret 2018 dari <https://mitbal.wordpress.com/2014/01/18/ml-apa-itu-machine-learning/>
- Munir. 2004. *Pengolahan Citra*. Bandung: Informatika
- Murni, Arniati. 1992. *Pengantar Pengolahan Citra*. Jakarta: PT. Elek Media Komputindo.
- Nasichuddin, Moch Ari. 2017. *Implementasi CNN untuk Klasifikasi Teks Menggunakan Tensorflow*. Diakses pada 7 Maret 2018 dari <https://medium.com/@arynas92/implementasi-cnn-untuk-klasifikasi-teks-menggunakan-tensorflow-3a720cc3afbc>
- Perkovic, Ljubomir. 2012. *Introduction to Computing Using Python: An Application Development Focus*.
- Pujoseno, Jimmy. 2018. *Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Alat Tulis*. Skripsi. Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Islam Indonesia.
- Ramesh Jain, Kasturi, Schunk. 1995. *Machine Vision*. McGraw-Hill.
- Rismiyati. 2016. *Implementasi Convolutional Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital*. Tesis. Program Studi S2 Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Gadjah Mada.

- Russa, Helder Filipe de Sausa. 2017. *Computer Vision: Object Recognition With Deep Learning Applied to Fashion Items Detection in Images*. Tesis. Faculdade de Economia Universidade Do Porto.
- Samuel, Arthur. 1959. *Machine Vision*. McGraw-Hill.
- Sena, Samuel. 2017. *Neural Network*. Diakses pada 7 Maret 2018 dari <https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac>
- Sena, Samuel. 2017. *Pengenalan Deep learning Part 7: Convolutional Neural Network (CNN)*. Diakses 3 Januari 2018 dari <https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94>.
- Sena, Samuel. 2017. *Pengenalan Deep learning Part 7: Convolutional Neural Network (CNN)*. Diakses 7 Maret 2018 dari <https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94>.
- Setiyono, Budi dan Zufar, Muhammad. 2016. *Convolutional Neural Network untuk Pengenalan Wajah Secara Real-Time*. Jurnal Sains dan Seni ITS Vol. 5 No.2 (2016) 2337-3520.
- Shapiro, L.G. dan Stockman, G.C., 2001. *Computer Vision*. 1<sup>st</sup> edition, Prentice Hall.
- Sharma, Sagar. 2017. *Activation Functions: Neural Networks*. Diakses dari <https://towardsdatascience.com/activation-functions-neural-networks1cbd9f8d91d6>
- Sitorus, Syahriol dkk. 2006. *Pengolahan Citra Digital*. Universitas Sumatera Utara.
- Suartika, I. Wayan., et.al. 2016. *Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101*. JURNAL TEKNIK ITS. Vol. 5, No. 1.
- Taufiq, Imam. 2018. *Deep Learning Untuk Deteksi Tanda Nomor Kendaraan Bermotor Menggunakan Algoritma Convolutional Neural Network Dengan Python Dan Tensorflow*. Skripsi. Program Studi Sistem Informasi Sekolah Tinggi Manajemen Informatika dan Komputer AKAKOM.

- Taufiq, Imam. 2018. *Tensorflow Custom Object Detection API*. Diakses pada tanggal 06 Maret 2018 pukul 06.00 WIB dari <https://imamdigmi.github.io/post/tensorflow-custom-object-detection/>
- Tensorflow. 2017. *Tensorflow Object Detection API*. Diakses pada tanggal 02 Februari 2018 dari <https://github.com/tensorflow/models/>
- TheMebel. 2016. *Ciri Khas Ukiran Jepara, Akar terkemukanya Jenis Ukiran yang Unik*. Diakses pada tanggal 12 April 2018 pukul 00.53 WIB dari <https://www.themebel.co.id/www.themebel.co.id/blog/ciri-khas-ukiran-jepara/>
- Tran, Dat. 2017. *How To Train Your Own Object Detector with Tensorflow's Object Detector API*. Diakses pada tanggal 01 Maret 2018 pukul 07.00 WIB dari <https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9>
- Tzotalin. 2017. *LabelImg*. Git Code. Github repository.
- Wikipedia. *Kursi*. Diakses pada tanggal 5 Maret 2018 pukul 10.00 WIB dari <https://id.wikipedia.org/wiki/Kursi>
- Wikipedia. *Meja*. Diakses pada tanggal 5 Maret 2018 pukul 10.05 WIB dari <https://id.wikipedia.org/wiki/Meja>
- Wikipedia. *Web Crawler*. Diakses pada 12 Februari 2018 pukul 15.00 WIB dari [https://en.wikipedia.org/wiki/Web\\_crawler](https://en.wikipedia.org/wiki/Web_crawler)
- Yanuar, Adi Hafiduddin. 2013. *Perbandingan Metode Gray Level Co-occurrence Matrix dengan Filter Gabor dalam Penentuan Jenis Kerang Berdasarkan Tekstur Cangkang*. Tugas Akhir. Jurusan Teknik Informatika Universitas Muhammadiyah Gresik.

## LAMPIRAN

### Lampiran 1 Script Javascript

```
// pull down jquery into the JavaScript console
var script = document.createElement('script');
script.src=
"https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(script);

// grab the URLs
var urls = $(''.rg_di .rg_meta').map(function(){return
JSON.parse($(this).text()).ou; });

// write the URLs to file (one per line)
var textToSave = urls.toArray().join('\n');
var hiddenElement = document.createElement('a');
hiddenElement.href = 'data:attachment/text,' +
encodeURIComponent(textToSave);
hiddenElement.target = '_blank';
hiddenElement.download = 'urls.txt';
hiddenElement.click();
```

## Lampiran 2 Script Python Crawling Data

```

# USAGE
# python download_images.py --urls urls.txt --output images/santa

# import the necessary packages
from imutils import paths
import argparse
import requests
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-u", "--urls", required=True,
                help="path to file containing image URLs")
ap.add_argument("-o", "--output", required=True,
                help="path to output directory of images")
args = vars(ap.parse_args())

# grab the list of URLs from the input file, then initialize the
# total number of images downloaded thus far
rows = open(args["urls"]).read().strip().split("\n")
total = 0

# loop the URLs
for url in rows:
    try:
        # try to download the image
        r = requests.get(url, timeout=60)

        # save the image to disk
        p = os.path.sep.join([args["output"], "{}.jpg".format(
            str(total).zfill(8))])
        f = open(p, "wb")
        f.write(r.content)
        f.close()

        # update the counter
        print("[INFO] downloaded: {}".format(p))
        total += 1

        # handle if any exceptions are thrown during the download process
        except:
            print("[INFO] error downloading {}...skipping".format(p))

# loop over the image paths we just downloaded
for imagePath in paths.list_images(args["output"]):
    # initialize if the image should be deleted or not
    delete = False

    # try to load the image

```

```
try:
    image = cv2.imread(imagePath)

    # if the image is `None` then we could not properly load it
    # from disk, so delete it
    if image is None:
        print("None")
        delete = True

    # if OpenCV cannot load the image then the image is likely
    # corrupt so we should delete it
    except:
        print("Except")
        delete = True

    # check to see if the image should be deleted
    if delete:
        print("[INFO] deleting {}".format(imagePath))
        os.remove(imagePath)
```

### Lampiran 3 Script Konversi Datasets Meta XML ke CSV

```

import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET

def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                    int(root.find('size')[0].text),
                    int(root.find('size')[1].text),
                    member[0].text,
                    int(member[4][0].text),
                    int(member[4][1].text),
                    int(member[4][2].text),
                    int(member[4][3].text)
                    )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin',
                  'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df

def main():
    for directory in ['train', 'test']:
        image_path = os.path.join(os.getcwd(),
                                  'annotations/{}'.format(directory))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv('data/{}_labels.csv'.format(directory), index=None)
        print('Successfully converted xml to csv.')

main()

```

#### Lampiran 4 Script Konversi Datasets CSV ke TFRecord

Usage:

```
# Create train data: python generate_tfrecord.py --type=train --
csv_input=data/train_labels.csv --output_path=data/train.record
# Create test data: python generate_tfrecord.py --type=test --
csv_input=data/test_labels.csv --output_path=data/test.record
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import
import os
import io
import pandas as pd
import tensorflow as tf
from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('type', '', 'Type of CSV input (train/test)')
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
FLAGS = flags.FLAGS

# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'Carving Chair':
        return 1
    elif row_label == 'Carving Table':
        return 2
    else:
        return 0

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
        encoded_jpg_io = io.BytesIO(encoded_jpg)
        image = Image.open(encoded_jpg_io)
        width, height = image.size

        filename = group.filename.encode('utf8')
        image_format = b'jpg'
        xmin = []
        xmax = []
```



```

ymins = []
ymaxs = []
classes_text = []
classes = []

for index, row in group.object.iterrows():
    xmin = row['xmin'] / width
    xmax = row['xmax'] / width
    ymin = row['ymin'] / height
    ymax = row['ymax'] / height
    classes_text.append(row['class'].encode('utf8'))
    classes.append(class_text_to_int(row['class']))

tf_example =
tf.train.Example(features=tf.train.Features(feature={
    'image/height': dataset_util.int64_feature(height),
    'image/width': dataset_util.int64_feature(width),
    'image/filename': dataset_util.bytes_feature(filename),
    'image/source_id': dataset_util.bytes_feature(filename),
    'image/encoded': dataset_util.bytes_feature(encoded_jpg),
    'image/format': dataset_util.bytes_feature(image_format),
    'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
    'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
    'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
    'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
    'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
    'image/object/class/label':
dataset_util.int64_list_feature(classes),
}))
return tf_example
def main():
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(),
'images/{}'.format(FLAGS.type))
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully created the TFRecords:
{}'.format(output_path))

if __name__ == '__main__':
    tf.app.run()

```

**Lampiran 5 Script Label Map**

```
item {  
  id: 1  
  name: 'Carving Chair'  
}
```

```
item {  
  id: 2  
  name: 'Carving Table'  
}
```

## Lampiran 6 Script Konfigurasi *Object Detection* Pelatihan Pipeline

```

model {
  ssd {
    num_classes: 2
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  anchor_generator {
    ssd_anchor_generator {
      num_layers: 6
      min_scale: 0.2
      max_scale: 0.95
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      aspect_ratios: 3.0
      aspect_ratios: 0.3333
    }
  }
  image_resizer {
    fixed_shape_resizer {
      height: 300
      width: 300
    }
  }
  box_predictor {
    convolutional_box_predictor {
      min_depth: 0
      max_depth: 0
      num_layers_before_predictor: 0
      use_dropout: false
      dropout_keep_probability: 0.8
    }
  }
}

```

```

kernel_size: 1
box_code_size: 4
apply_sigmoid_to_scores: false
conv_hyperparams {
  activation: RELU_6,
  regularizer {
    l2_regularizer {
      weight: 0.00004
    }
  }
  initializer {
    truncated_normal_initializer {
      stddev: 0.03
      mean: 0.0
    }
  }
  batch_norm {
    train: true,
    scale: true,
    center: true,
    decay: 0.9997,
    epsilon: 0.001,
  }
}
}
feature_extractor {
  type: 'ssd_mobilenet_v1'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
      l2_regularizer {
        weight: 0.00004
      }
    }
  }
  initializer {
    truncated_normal_initializer {
      stddev: 0.03
      mean: 0.0
    }
  }
  batch_norm {
    train: true,
    scale: true,
    center: true,
    decay: 0.9997,
    epsilon: 0.001,
  }
}
}

```

```

}
loss {
  classification_loss {
    weighted_sigmoid {
      anchorwise_output: true
    }
  }
  localization_loss {
    weighted_smooth_l1 {
      anchorwise_output: true
    }
  }
  hard_example_miner {
    num_hard_examples: 3000
    iou_threshold: 0.99
    loss_type: CLASSIFICATION
    max_negatives_per_positive: 3
    min_negatives_per_image: 0
  }
  classification_weight: 1.0
  localization_weight: 1.0
}
normalize_loss_by_num_matches: true
post_processing {
  batch_non_max_suppression {
    score_threshold: 1e-8
    iou_threshold: 0.6
    max_detections_per_class: 100
    max_total_detections: 100
  }
  score_converter: SIGMOID
}
}
}
train_config: {
  batch_size: 2
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.004
          decay_steps: 800720
          decay_factor: 0.95
        }
      }
      momentum_optimizer_value: 0.9
      decay: 0.9
      epsilon: 1.0
    }
  }
}
}

```

```
    fine_tune_checkpoint:
"ssd_mobilenet_v1_coco_2017_11_17/model.ckpt"
    from_detection_checkpoint: true
    num_steps: 250000
    data_augmentation_options {
      random_horizontal_flip {
      }
    }
    data_augmentation_options {
      ssd_random_crop {
      }
    }
  }
train_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/charving_label_map.pbtxt"
}
eval_config: {
  num_examples: 30
  max_evals: 10
}
eval_input_reader: {
  tf_record_input_reader {
    input_path: "data/test.record"
  }
  label_map_path: "data/charving_label_map.pbtxt"
  shuffle: false
  num_readers: 1
  num_epochs: 1
}
```

## Lampiran 7 Training Neural Network

```

import functools
import json
import os
import tensorflow as tf

from object_detection import trainer
from object_detection.builders import input_reader_builder
from object_detection.builders import model_builder
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master
to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy
per worker.')
flags.DEFINE_boolean('clone_on_cpu', False,
'Force clones to be deployed on CPU. Note that even if '
'set to False (allowing ops to run on gpu), some ops may '
'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of
worker+trainer '
'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
'Number of parameter server tasks. If None, does not use '
'a parameter server.')
flags.DEFINE_string('train_dir', '',
'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
'Path to a train_pb2.TrainConfig config file.')
flags.DEFINE_string('input_config_path', '',
'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS

def main(_):
    assert FLAGS.train_dir, '`train_dir` is missing.'
    if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
    if FLAGS.pipeline_config_path:

```

```

configs = config_util.get_configs_from_pipeline_file(
    FLAGS.pipeline_config_path)
if FLAGS.task == 0:
    tf.gfile.Copy(FLAGS.pipeline_config_path,
        os.path.join(FLAGS.train_dir, 'pipeline.config'),
        overwrite=True)
else:
    configs = config_util.get_configs_from_multiple_files(
        model_config_path=FLAGS.model_config_path,
        train_config_path=FLAGS.train_config_path,
        train_input_config_path=FLAGS.input_config_path)
    if FLAGS.task == 0:
        for name, config in [('model.config', FLAGS.model_config_path),
            ('train.config', FLAGS.train_config_path),
            ('input.config', FLAGS.input_config_path)]:
            tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                overwrite=True)

model_config = configs['model']
train_config = configs['train_config']
input_config = configs['train_input_config']

model_fn = functools.partial(
    model_builder.build,
    model_config=model_config,
    is_training=True)

create_input_dict_fn = functools.partial(
    input_reader_builder.build, input_config)

env = json.loads(os.environ.get('TF_CONFIG', '{}'))
cluster_data = env.get('cluster', None)
cluster = tf.train.ClusterSpec(cluster_data) if cluster_data else
None
task_data = env.get('task', None) or {'type': 'master', 'index':
0}
task_info = type('TaskSpec', (object,)), task_data

# Parameters for a single worker.
ps_tasks = 0
worker_replicas = 1
worker_job_name = 'lonely_worker'
task = 0
is_chief = True
master = ''

if cluster_data and 'worker' in cluster_data:
    # Number of total worker replicas include "worker"s and the
    "master".
    worker_replicas = len(cluster_data['worker']) + 1
    if cluster_data and 'ps' in cluster_data:

```



```
ps_tasks = len(cluster_data['ps'])

if worker_replicas > 1 and ps_tasks < 1:
    raise ValueError('At least 1 ps task is needed for distributed
training.')
```

```
if worker_replicas >= 1 and ps_tasks > 0:
    # Set up distributed training.
    server = tf.train.Server(tf.train.ClusterSpec(cluster),
protocol='grpc',
job_name=task_info.type,
task_index=task_info.index)
    if task_info.type == 'ps':
        server.join()
    return
```

```
worker_job_name = '%s/task:%d' % (task_info.type, task_info.index)
task = task_info.index
is_chief = (task_info.type == 'master')
master = server.target
```

```
trainer.train(create_input_dict_fn, model_fn, train_config,
master, task,
FLAGS.num_clones, worker_replicas, FLAGS.clone_on_cpu, ps_tasks,
worker_job_name, is_chief, FLAGS.train_dir)
```

```
if __name__ == '__main__':
    tf.app.run()
```

## Lampiran 8 Script Export Graph Model

```

import tensorflow as tf
from google.protobuf import text_format
from object_detection import exporter
from object_detection.protos import pipeline_pb2

slim = tf.contrib.slim
flags = tf.app.flags
flags.DEFINE_string('input_type', 'image_tensor', 'Type of input
node. Can be 'one of [image_tensor, encoded_image_string_tensor,
''tf_example'']')
flags.DEFINE_string('input_shape', None,
                    'If input_type is image_tensor, this can
explicitly set the shape of this input tensor to a fixed size.
The dimensions are to be provided as a comma-separated list
of integers. A value of -1 can be used for unknown
dimensions. If not specified, for an image_tensor, the
default shape will be partially specified as
[None, None, None, 3].')
flags.DEFINE_string('pipeline_config_path', None,
                    'Path to a pipeline_pb2.TrainEvalPipelineConfig
config file.')
flags.DEFINE_string('trained_checkpoint_prefix', None,
                    'Path to trained checkpoint, typically of the
form path/to/model.ckpt')
flags.DEFINE_string('output_directory', None, 'Path to write
outputs.')
tf.app.flags.mark_flag_as_required('pipeline_config_path')
tf.app.flags.mark_flag_as_required('trained_checkpoint_prefix')
tf.app.flags.mark_flag_as_required('output_directory')
FLAGS = flags.FLAGS
def main():
    pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
    with tf.gfile.GFile(FLAGS.pipeline_config_path, 'r') as f:
        text_format.Merge(f.read(), pipeline_config)
    if FLAGS.input_shape:
        input_shape = [
            int(dim) if dim != '-1' else None
            for dim in FLAGS.input_shape.split(',')
        ]
    else:
        input_shape = None
    exporter.export_inference_graph(FLAGS.input_type,
    pipeline_config,
                                FLAGS.trained_checkpoint_prefix,
                                FLAGS.output_directory,
    input_shape)
if __name__ == '__main__':
    tf.app.run()

```

## Lampiran 9 Script Deteksi Meja dan Kursi

```

import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

if tf.__version__ < '1.4.0':
    raise ImportError('Please upgrade your tensorflow installation
to v1.4.* or later!')

# This is needed to display the images.
%matplotlib tk
# This is needed since the notebook is stored in the
object_detection folder.
sys.path.append('/D:/Cara/new/models-master/research') #point to
your tensorflow
sys.path.append('/D:/Cara/new/models-master/research/slim') #point
ot your slim

from utils import label_map_util
from utils import visualization_utils as vis_util

# What model to download.
MODEL_NAME = 'aku'

# Path to frozen detection graph. This is the actual model that is
used for the object detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each
box.
PATH_TO_LABELS = os.path.join('data', 'charving_label_map.pbtxt')

NUM_CLASSES = 2

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()

```

```

    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)

# For the sake of simplicity we will use only 2 images:
# image1.jpg
# image2.jpg
# If you want to test the code with your images, just add path to
the images to the TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(4, 9) ]

# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)

with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        # Definite input and output Tensors for detection_graph
        image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')
        # Each box represents a part of the image where a particular
object was detected.
        detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
        # Each score represent how level of confidence for each of the
objects.
        # Score is shown on the result image, together with the class
label.
        detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
        detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')
        num_detections =
detection_graph.get_tensor_by_name('num_detections:0')
        for image_path in TEST_IMAGE_PATHS:
            image = Image.open(image_path)
            # the array based representation of the image will be used
later in order to prepare the

```

```
# result image with boxes and labels on it.
image_np = load_image_into_numpy_array(image)
# Expand dimensions since the model expects images to have
shape: [1, None, None, 3]
image_np_expanded = np.expand_dims(image_np, axis=0)
# Actual detection.
(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes,
num_detections],
    feed_dict={image_tensor: image_np_expanded})
# Visualization of the results of a detection.
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8)
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)
```

## Lampiran 10 Script Video

```

import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
import cv2

if tf.__version__ < '1.4.0':raise ImportError('Please upgrade your
tensorflow installation to v1.4.* or later!')

# Video Setup
# cap = cv2.VideoCapture(0)
cap = cv2.VideoCapture('./videol.mp4')

# Environment setup
sys.path.append("../")

# Object detection imports
# Here are the imports from the object detection module.
from utils import label_map_util
from utils import visualization_utils as vis_util

# Model preparation
MODEL_NAME = 'carving_table_model'
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = os.path.join('data', 'carving_table_map.pbtxt')
NUM_CLASSES = 1

# Load a (frozen) Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)

```

```

category_index = label_map_util.create_category_index(categories)

with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
#Definite input and output Tensors for detection_graph
    image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')

#Each box represents a part of the image where a particular object
was detected.
    detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')

# Each score represent how level of confidence for each of the
objects.
# Score is shown on the result image, together with the class
label.
    detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
    detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')
    num_detections =
detection_graph.get_tensor_by_name('num_detections:0')
    while(cap.isOpened()):
        ret, image_np = cap.read()

# Expand dimensions since the model expects images to have shape:
[1, None, None, 3]
image_np_expanded = np.expand_dims(image_np, axis=0)

# Actual detection.
(boxes, scores, classes, num) = sess.run([detection_boxes,
detection_scores, detection_classes, num_detections],
feed_dict={image_tensor: image_np_expanded})

# Visualization of the results of a detection.
    vis_util.visualize_boxes_and_labels_on_image_array(
        image_np,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8)

cv2.imshow('Carving Detection', cv2.resize(image_np, (1000,800)))
if cv2.waitKey(25) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    break

```