Available online at https://ejournal.almaata.ac.id/index.php/IJUBI
Indonesian Journal of Business Intelligence
Volume 4 | Issue 1 | June (2021)
IJUBI
Indonesian Journal ---- of ---- Business Intelligence

ANALISA PERFORMA ARSITEKTUR MOBILENETV1 DAN RESNET MENGGUNAKAN META-LEARNING DALAM MENDETEKSI OBJEK HEWAN KUCING

Faiz Octa Reynaldi[1], Omar Pahlevi[2], Indah Suryani[3]
[13]Teknik Informatika, Fakultas Ilmu Komputer, Sekolah Tinggi Manajemen Informatika dan Komputer Nusa Mandiri,
[2]Sistem Informasi, Fakultas Teknik dan Informatika, Universitas Bina Sarana Informatika.
faizoctareynaldi@gmail.com , omar.opi@bsi.ac.id , indah.ihy@nusamandiri.ac.id
[13] Jl. Jatiwaringin No. 02, Daerah Khusus Ibukota Jakarta 13620
[2]Jl. Kramat Raya No. 98, Daerah Khusus Ibukota Jakarta 10450

Keywords:

Object Detection, Transfer Learning, Cloud Computing, Few-Shot Learning, Cat

Abstract

Object Detection has several obstacles during the training process such as the amount of data that must be trained, takes a long time to be trained, and so on. In this study, the researchers compared the accuracy and average loss training of the MobileNetV1 SSD architecture and ResNet SSD using the Pre-Trained model with the Few-Shot Learning method using Hold-Out Cross Validation to detect Black Cat Animal Objects and White Cat Animal Objects by collecting data automatically real data from the Jakarta Vet Shop observation method and only requires a small amount of data for the training process. This research was conducted by using Cloud Computing such as Google Colab as a medium to compare the architectural accuracy of the MobileNetV1 SSD and the ResNet SSD. 97.9% for white cats while for SSD MobileNetV1 it has a mean score of 99.66666667% on black cats and 78.733% on white cats. Then SD MobileNetV1 has a greater Train Loss with an average value of 0.003923 for Black Cats and an average value of 0.0059 for White Cats when compared to ResNet's SSD with an average value of 0.030263 for Black Cats and an average value of 0.00413 for White Cats.

Kata Kunci:

Object Detection, Transfer Learning, Cloud Computing, Few-Shot Learning, Hewan Kucing

Abstrak

Object Detection memiliki beberapa kendala saat proses *training* seperti banyaknya data yang harus dilatih, menggunakan waktu cukup lama untuk dilatih. Pada penelitian ini, peneliti melakukan komparasi akurasi *dan average loss training* arsitektur *SSD MobileNetV1* dan *SSD ResNet* menggunakan *Pre-Trained model* dengan metode *Few-Shot Learning* menggunakan *Hold-Out Cross Validation* untuk mendeteksi Objek Hewan Kucing Hitam dan Objek Hewan Kucing Putih dengan pengambilan data secara riil dari metode observasi Jakarta *Vet Shop* dan hanya membutuhkan sedikit data untuk dilakukannya proses *training*. Penelitian ini dilakukan dengan cara menggunakan *Cloud Computing* seperti *Google Colab* sebagai media untuk membandingkan akurasi arsitektur *SSD MobileNetV1* dan *SSD ResNet*. Hasil analisa dalam penelitian ini adalah *SSD ResNet* memiliki akurasi yang tinggi dengan nilai rata-rata 100% pada kucing hitam dan nilai rata-rata 97.9% pada kucing putih sementara untuk *SSD MobileNetV1* memiliki nilai rata-rata 99.66666667% pada kucing hitam dan 78.733% pada kucing putih. Kemudian *SSD MobileNetV1* memiliki *Train Loss* lebih besar dengan nilai rata-rata 0.003923 pada Kucing Hitam dan nilai rata-rata 0.0059 Kucing Putih jika dibandingkan dengan *SSD ResNet* dengan nilai rata-rata 0.030263 pada Kucing Hitam dan nilai rata-rata 0.00413 pada Kucing Putih.

Pendahuluan

Kebutuhan manusia akan sistem teriring dengan perkembangan pengetahuan dalam sistem komputer [1]. Dimana penggunaan teknologi informasi saat ini berkembang sangat pesat. Dengan adanya teknologi informasi diharapkan berbagai praktisi dapat terbantu dalam segala bidang. Hal ini dikarenakan teknologi informasi dapat menyimpan data hingga mengolahnya menjadi suatu hasil yang diinginkan pembuatnya [2].

Berbagai macam teknik pembelajaran dan optimasi melalui konsep *machine learning* dan *swarm intelligence* dapat memudahkan bagi pengguna di bidang teknologi kecerdasan buatan. Adapun pengertian kecerdasan buatan merupakan suatu mesin atau program yang memiliki kecerdasan didalamnya yang berguna menyelesaikan suatu pekerjaan [3]

Object detection telah menjadi topik masalah di keduanya bidang penginderaan jauh dan visi komputer. Ini umumnya didefinisikan sebagai mengidentifikasi lokasi objek target dalam gambar input serta mengenali kategori objek. Deteksi objek otomatis telah banyak digunakan di banyak aplikasi dunia nyata, seperti deteksi bahaya, pemantauan lingkungan, deteksi perubahan, perencanaan kota, dan lain-lain [4].

Lalu *object detection* memiliki kendala saat memasuki fase *preparation*, diantaranya membutuhkan banyaknya data pada sebuah *dataset*. Selain itu diperlukan metode seperti *train* dan *test splitting* seperti *cross-validation* [5]. Dengan metode tersebut, saat melakukan proses *training* membutuhkan waktu yang cukup lama.

Penelitian ini terkait dengan penelitian yang dilakukan oleh [6]. Dalam penelitian ini membahas mengenai arsitektur *residual-network* baru, *Residual networks of Residual networks* (RoR) untuk menggali kemampuan optimalisasi jaringan residual. RoR menggantikan optimalisasi pemetaan sisa dari pemetaan sisa untuk mengoptimalkan pemetaan residu asli. Secara khusus, RoR menambahkan koneksi pintas level-bijaksana pada jaringan residual asli untuk mempromosikan kemampuan pembelajaran jaringan sisa. Lebih penting lagi, RoR dapat diterapkan ke berbagai jenis jaringan residual (*ResNets*, *Pre-ResNets* dan WRN) dan secara signifikan meningkatkan kinerjanya. Dimana hasil eksperimen menunjukkan keefektifan dan keserbagunaan RoR, sehingga mencapai kinerja terbaik di semua struktur serupa jaringan sisa. Model RoR-3-WRN58-4 + SSD mencapai hasil mutakhir baru pada CIFAR-10, CIFAR-100 dan SVHN, dengan kesalahan uji masing-masing 3,77%, 19,73% dan 1,59%. Model RoR-3 juga mencapai hasil mutakhir dibandingkan dengan *ResNets* pada kumpulan data *ImageNet*.

Lalu pada penelitian yang dilakukan oleh [7], dimana membahas mengenai yang telah dilakukan, berhasil dibuat aplikasi untuk mendeteksi objek dengan menggunakan *TensorFlow Object Detection* API dengan memanfaatkan *SSD Mobilenet V2* sebagai model pra-terlatih. Program dapat mendeteksi 5 kategori kelas objek, yaitu Camera, Handphone, Headphone, Laptop, dan Mouse. Aplikasi dapat menampilkan tingkat pengukuran akurasi masing-masing objek. Hasil pengujian yang dilakukan pada 50 test set dengan jumlah masing-masing kelas objek adalah 10 gambar. Gambar objek Camera mendapatkan rata-rata presentase sebesar 99%, Handphone sebesar 89.1%, Headphone sebesar 89.1%, Laptop sebesar 89.1%, dan Mouse sebesar 98.8%. sehingga diperoleh rata-rata akurasi keberhasilan pendeteksian aplikasi deteksi objek adalah 93.02%.

Kemudian pada penelitian [8], membahas mengenai pengolahan *dataset* dan *training* pada *Machine Learning* dilakukan pada *server-side* dan *prediction* dilakukan *client-side* menggunakan *browser* dengan metode pengembangan sistem *Rapid Application Development* dan membutuhkan banyak *dataset* untuk membuat suatu *Pre-Trained Model MobileNet*. Bahasa program yang digunakan adalah *Python* dan menggunakan *tools* seperti *Jupyter Notebook*, *Tensorflow Keras* dan *Pillow*. Pada hasil akhir dari penelitian ini adalah dapat memudahkan dari sisi *client-user* untuk melakukan membuat model klasifikasi tanpa mengunduh dependensi dan mengonversi model terlebih dahulu,

sehingga pengguna dapat melakukan klasifikasi secara dinamis.

Pada penelitian yang dibuat oleh peneliti ini memfokuskan untuk mengefisiensi metode sebelumnya dengan metode *meta-learning* berbasis *few shot*. Adapun *meta-learning* adalah metode yang menggunakan *optimization algorithm* [9], dimana untuk melatih sebuah *dataset* hanya membutuhkan data yang lebih sedikit tetapi memiliki hasil yang lebih akurat [10].

Komparasi ke-dua Arsitektur antara *SSD MobileNetV1 dan SSD ResNet* memiliki metode *training model*, perhitungan dan *tunning* akurasi yang berbeda, perbedaan inilah yang membuat peneliti menjadi tertarik dengan mempelajari dan mengamati lebih dalam mengenai tingkat akurasi identifikasi obyek hewan kucing.

Landasan Teori

A. *Image Processing*

Image processing merupakan kegiatan komputer dalam hal manipulasi dan analisis suatu informasi gambar. Adapun informasi gambar yang dimaksud dengan adalah gambar visual dalam dua dimensi, dimana *image processing* merupakan operasi yang dapat dilakukan untuk memperbaiki, menganalisis, atau mengubah suatu gambar [11].

B. *Machine Learning*

Machine Learning (pembelajaran mesin) adalah salah satu dari beberapa cabang Articial Intelligence (kecerdasan buatan) dimana komputer dapat mengakses data yang sudah ada sesuai dengan perintahnya sendiri [12].

C. *Meta-Learning*

Meta-Learning juga dikenal sebagai *learning how to learn*. Paradigma pembelajaran potensial yang baru-baru ini muncul dapat menyerap informasi dari satu tugas dan menggeneralisasi informasi tersebut menjadi tugas yang sulit dilihat. *Meta-Learning* pada umumnya berkerja dengan cara melibatkan penyalinan parameter jaringan pertama ke dalam parameter jaringan kedua atau ke pengoptimal [13].

D. *Python*

Python merupakan bahasa pemrograman yang memiliki keunggulan antara lain *readability*, efisien, multifungsi, interoperabilitas, dan memiliki dukungan komunitas yang memadai kesalahan, *bug* atau *error l*ainnya [14].

E. *Tensorflow*

TensorFlow merupakan salah satu *framework machine learning* yang dapat digunakan dalam environment yang heterogeneous. Dimana dapat melakukan eksperimen model *deep learning*, melatih model pada *dataset* yang berukuran besar, dan membuatnya layak diproduksi [15].

F. *Google Colab*

Google Colab adalah *coding environment* berbasis bahasa pemrograman *python*. Formatnya adalah *notebook* hampir serupa dengan *Jupyter notebook* berbasis *cloud computing* [16].

G. *MobileNet*

MobileNet merupakan salah satu arsitektur *convolutional neural network* yang dapat digunakan untuk mengatasi permintaan sumber daya komputasi yang berlebihan [8].

H. *Residual Neural Network* (*ResNet*)

Residual Neural Network atau yang biasa dikenal dengan nama ResNet adalah satu arsitektur yang popular didalam mendeteksi suatu objek [17].

I. *Few-Shot Learning*

Few-Shot Learning (FSL) adalah jenis masalah pembelajaran mesin (ditentukan oleh *experience*, *task* dan *performance*), di mana *experience* hanya berisi sejumlah contoh dengan informasi yang diawasi untuk target *task* [18].

J. *Common Objects in Context (COCO)*

Common Objects in Context atau biasa disingkat dengan COCO adalah *dataset* dengan skala besar untuk deteksi objek dan segmentasi objek. Pada tahun 2015 COCO *dataset* telah memiliki sebanyak 165.482 *train*, 82.208 *validation* dan 81.434 *test* atau sebanyak 50% dalam *train*, 25% dalam *validation* dan 25% dalam *test* [19].

K. *Domestic shord-haired cat*

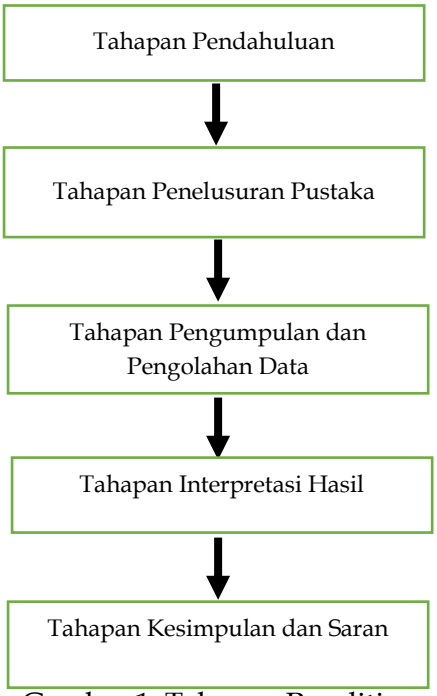
Pada penelitian ini, peneliti menggunakan obyek penelitian kucing dengan ras *Domestic shord-haired cat*. Dimana ras ini merupakan merupakan jenis kucing berbulu pendek di luar semua ras kucing yang ada. Tipe ras kucing ini mayoritas muncul akibat kecelakaan kawin

silang, diakibatkan perkawinan silang yang tidak disengaja dari dua ras kucing. Kemudian kucing-kucing ini kawin dengan kucing lainnya (baik kucing yang memiliki ras tertentu atau pun kucing sejenis), sehingga menghasilkan berbagai kucing yang tidak mempunyai ras tertentu [20].

Metode

A. Tahapan Penelitian

Tahap penelitian meliputi langkah implementasi dari awal. Langkah-langkahnya adalah sebagai berikut:



Gambar 1. Tahapan Penelitian

1. Tahapan Pendahuluan

Pada tahapan ini dilakukan studi literatur dan penelitian ke lapangan. Studi pustaka dilakukan untuk mempelajari secara teori dan menentukan metode yang digunakan dalam metode penelitian. Sedangkan penelitian lapangan untuk mempelajari hubungan antara metode penelitian dengan objek penelitian pada Jakarta Vet Shop Jakarta Pusat.

2. Tahapan Penelusuran Pustaka

Peneliti mencari bahan referensi yang berkaitan dengan penelitian sebelumnya yang berkaitan dengan tujuan penelitian, untuk mengetahui kontribusi penelitian dan melakukan penelitian kepustakaan untuk penelitian teoritis penelitian ini.

3. Tahapan Pengumpulan dan Pengolahan Data

Peneliti mengumpulkan data yang diperlukan sebagai bahan pemecahan masalah. Setelah itu akan dilakukan pengolahan data untuk mendapatkan interpretasi hasil penelitian

4. Tahapan Intepretasi Hasil

Peneliti menganalisis hasil pengolahan data berdasarkan hasil penelitian dan penelitian teoritis yang ada.

5. Tahapan Kesimpulan dan Saran

Pada tahap ini peneliti menarik kesimpulan atas hasil penelitian berdasarkan pengumpulan dan pengolahan data. Kesimpulan ini diambil dari hasil penelitian yang ada. Berdasarkan kesimpulan tersebut, peneliti mengajukan saran-saran yang relevan untuk proses terkait dengan tujuan penelitian agar dapat memberikan hasil yang lebih baik di masa yang akan datang.

B. Metode Pengumpulan Data

Berikut ini adalah metode pengumpulan data ada dua yaitu Penelitian Lapangan dan Penelitian Kepustakaan, berikut adalah penjelasan dari Penelitian Lapangan dan Penelitian Kepustakaan:

1. Penelitian Lapangan

Penelitian di lapangan merupakan kajian langsung terhadap objek penelitian, didalam penelitian ini objek yang diambil gambar secara ril untuk dilatih oleh mesin adalah hewan Kucing. Untuk memperkuat penelitian, peneliti melakukan observasi langsung pada Jakarta Vet Shop yang berada di Jalan Cempaka Putih Tengah XXX No. 5 RT. 3/ RW. 8, Jakarta Pusat.

2. Penelitian Kepustakaan

Penelitian Kepustakaan adalah teknik yang didasarkan pada literatur digunakan dalam metode penelitian. Selain itu, bentuk pencarian dokumen lain adalah jurnal penelitian, pertanyaan dan metodenya mirip dengan referensi penelitian.

Hasil dan Pembahasan

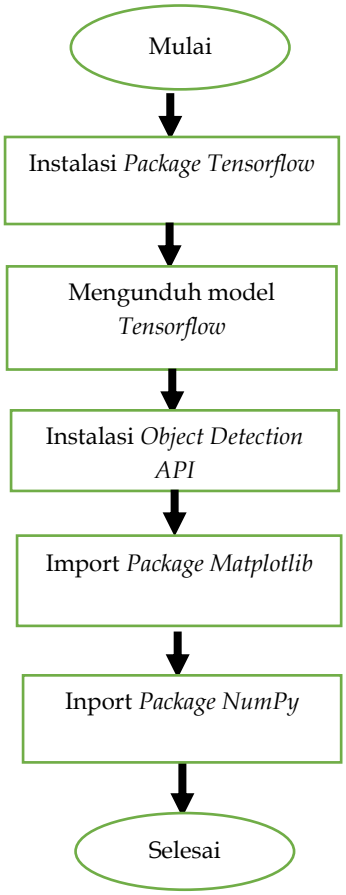
A. Pengumpulan Data

Pada penelitian ini menggunakan data primer dan sekunder. Dimana data primer diperoleh dari hasil observasi di Jakarta Vet Shop sebanyak lima objek hewan Kucing Putih dengan nama label kucingputih dan lima objek

hewan Kucing Hitam dengan nama label kucinghitam untuk melakukan *Training Model*, sementara untuk data sekunder sendiri didapatkan peneliti dengan menggunakan *Pre-Trained Model SSD MobileNetV1* dan *SSD ResNet*.

B. Alur Proses *Installasi Package Python*

Untuk mempermudah proses instalasi *Package Python*, peneliti membuat alur prosesnya sebagai berikut:



Gambar 2. Alur Proses *Installasi Package Python*

1. Instalasi *Package Tensorflow*

Langkah pertama didalam Proses Instalasi *Package Python* adalah dengan menginstal *Package Tensorflow* versi 2.2.

```
[ ]  !pip install -U --pre tensorflow=="2.2.0"
```

Gambar 3. Instalasi *Package Tensorflow*

2. Mengunduh model *Tensorflow*

Langkah kedua adalah mengunduh model *Tensorflow* melalui *Repository* yang tersedia melalui *Github*.

```
[ ]  import os
     import pathlib

     # Clone the tensorflow models repository if it doesn't already exist
     if "models" in pathlib.Path.cwd().parts:
       while "models" in pathlib.Path.cwd().parts:
         os.chdir('..')
     elif not pathlib.Path('models').exists():
       !git clone --depth 1 https://github.com/tensorflow/models
```

Gambar 4. Proses Unduh Model *Tensorflow*

3. Instalasi *Object Detection API*

Langkah ketiga adalah Instalasi *Object Detection API*, langkah ini dapat dilakukan setelah proses mengunduh model *Tensorflow* berhasil.

```
# Install the Object Detection API
%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

Gambar 5. Instalasi *Object Detection API*

4. *Import Package Matplotlib*

Pada langkah keempat adalah *Import Package Matplotlib*, pada umumnya *Matplotlib* sudah terinstal secara langsung jika sudah menginstal *Tensorflow* terlebih dahulu.

```
import matplotlib
import matplotlib.pyplot as plt

import os
import random
import io
import imageio
import glob
import scipy.misc
import numpy as np
from six import BytesIO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display, Javascript
from IPython.display import Image as IPyImage

import tensorflow as tf

from object_detection.utils import label_map_util
from object_detection.utils import config_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.utils import colab_utils
from object_detection.builders import model_builder

%matplotlib inline
```

Gambar 6. *Import Package Matplotlib*

5. *Import Package NumPy Array*

Langkah terakhir dalam proses Instalasi *Package Python* adalah melalukan *Import Package NumPy Array*. Sama halnya seperti *Matplotlib*, *NumPy* umumnya sudah terinstal jika *Tensorflow* sudah terinstal terlebih dahulu.

```
def load_image_into_numpy_array(path):
  """Load an image from file into a numpy array.

  """
  img_data = tf.io.gfile.GFile(path, 'rb').read()
  image = Image.open(BytesIO(img_data))
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)

def plot_detections(image_np,
                    boxes,
                    classes,
                    scores,
                    category_index,
                    figsize=(12, 16),
                    image_name=None):

  image_np_with_annotations = image_np.copy()
  viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_annotations,
      boxes,
      classes,
      scores,
      category_index,
      use_normalized_coordinates=True,
      min_score_thresh=0.8)
  if image_name:
    plt.imsave(image_name, image_np_with_annotations)
  else:
    plt.imshow(image_np_with_annotations)
```

Gambar 7. *Import Package NumPy Array*

C. Proses memberikan label pada Objek Hewan Kucing

Pada tahap ini dilakukan proses memberikan label pada Objek Hewan Kucing dengan cara membuat anotasi label sebanyak 5 (lima) gambar Hewan Kucing. Ada beberapa tahapan dalam melakukan Proses ini yaitu *Load Data Image* dan Anotasi Objek Hewan Kucing.

1.*Load Data Image*

Langkah pertama adalah dengan cara *Load Data Image* salin 5 (lima) Objek Hewan Kucing Putih dan Hewan Kucing Hitam kedalam direktori *models/research/object_detection/test_images/kucingputih/train/* untuk Objek Hewan Kucing Putih dan *models/research/object_detection/test_images/*kucing hitam*/train/* untuk Objek Hewan Kucing Hitam.

```
# Load images and visualize
train_image_dir = 'models/research/object_detection/test_images/kucingputih/train/'
train_images_np = []
for i in range(1, 6):
  image_path = os.path.join(train_image_dir, 'kucing' + str(i) + '.jpg')
  train_images_np.append(load_image_into_numpy_array(image_path))

plt.rcParams['axes.grid'] = True
plt.rcParams['xtick.labelsize'] = True
plt.rcParams['ytick.labelsize'] = True
plt.rcParams['xtick.top'] = True
plt.rcParams['xtick.bottom'] = True
plt.rcParams['ytick.left'] = True
plt.rcParams['ytick.right'] = True
plt.rcParams['figure.figsize'] = [14, 7]

for idx, train_image_np in enumerate(train_images_np):
  plt.subplot(2, 8, idx+1)
  plt.imshow(train_image_np)
plt.show()
```

Gambar 8. Import Objek Hewan Kucing Putih

```
# Load images and visualize
train_image_dir = 'models/research/object_detection/test_images/kucinghitam/train/'
train_images_np = []
for i in range(1, 6):
  image_path = os.path.join(train_image_dir, 'kucing' + str(i) + '.jpg')
  train_images_np.append(load_image_into_numpy_array(image_path))

plt.rcParams['axes.grid'] = True
plt.rcParams['xtick.labelsize'] = True
plt.rcParams['ytick.labelsize'] = True
plt.rcParams['xtick.top'] = True
plt.rcParams['xtick.bottom'] = True
plt.rcParams['ytick.left'] = True
plt.rcParams['ytick.right'] = True
plt.rcParams['figure.figsize'] = [14, 7]

for idx, train_image_np in enumerate(train_images_np):
  plt.subplot(2, 8, idx+1)
  plt.imshow(train_image_np)
plt.show()
```

Gambar 9. Import Objek Hewan Kucing Hitam

2.Anotasi Objek Hewan Kucing

Pada langkah anotasi objek hewan Kucing yaitu memberikan label pada objek hewan Kucing Hitam dan Hewan Kucing Putih yang diambil secara riil pada Jakarta Vet Shop menggunakan *bounding box* dengan cara menjalankan perintah pada bahasa pemrograman *python* lalu berikan label menggunakan *bounding box* pada objek hewan Kucing Hitam dan objek hewan Kucing Putih.

```
[ ] gt_boxes = []
    colab_utils.annotate(train_images_np, box_storage_pointer=gt_boxes)

    '--boxes array populated--'
```

Gambar 10. Menjalankan Perintah Memberikan Label

Berikut ini tampilan memberikan label pada kucinghitam dan label kucingputih.

Gambar 11. Memberikan Label kucinghitam

Gambar 12. Memberikan Label kucingputih

D. *Training* Model

Training model pada Arsitektur *SSD MobileNetV1* dengan menggunakan kedua objek Hewan Kucing. *Training* tahap pertama adalah dengan melakukan *sample* pada objek hewan Kucing Hitam, pada *batch* 100 mendapatkan *total Loss* sebesar 0.095, pada *batch* 300 mendapatkan t*otal Loss* sebesar 0.0016 dan 0.00067 pada *batch* 500.

Sedangkan *training model* pada Arsitektur *SSD ResNet* dengan menggunakan kedua objek hewan Kucing. *Training* tahap pertama adalah dengan melakukan *sample* pada objek hewan Kucing Hitam, pada batch 100 mendapatkan *total Loss* sebesar 0.0054, pada *batch* 300 mendapatkan *total Loss* sebesar 0.00039 dan 0.085 pada *batch* 500.

E. Menguji Model

Pada tahap berikutnya adalah menguji model, tahapan ini adalah menguji tingkat akurasi dari Objek Hewan Kucing yang sudah di *Training* Model sebelumnya. Untuk menguji tingkat akurasi peneliti menggunakan lima gambar kucingputih dan lima gambar kucinghitam untuk dilakukan pengujian model.

Gambar 5. Pengujian Gambar kucinghitam

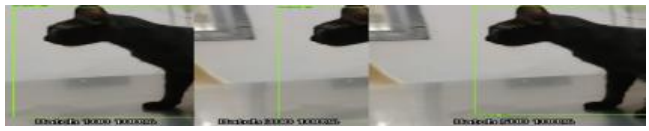
Gambar 6. Pengujian Gambar kucingputih

Berikut adalah pengujian menggunakan gambar kucinghitam dan kucingputih sebanyak 100, 300 dan 500 *batch* menggunakan *SSD MobileNetV1*.

Gambar 7. Akurasi *Batch* 100, 300 dan 500 kucinghitam menggunakan *SSD MobileNetV1*

Gambar 8. Akurasi *Batch* 100, 300 dan 500 kucingputih menggunakan *SSD MobileNetV1*

Berikut adalah pengujian menggunakan kucinghitam dan kucingputih sebanyak 100, 300 dan 500 *batch* menggunakan *SSD ResNet*.



Gambar 9. Akurasi *Batch* 100, 300 dan 500 kucinghitam menggunakan *SSD ResNet*

Gambar 10. Akurasi *Batch* 100, 300 dan 500 kucingputih menggunakan *SSD ResNet*

F. Evaluasi model

Evaluasi model *train loss* adalah mengevaluasi jumlah *loss* saat terjadinya proses *training* model dalam penelitian ini, berikut adalah Tabel Evaluasi Model *Train Loss*.

Tabel 1. Evaluasi Model *Train Loss*

Label	SSD MobileNetV1			SSD ResNet		
Jenis	100 batch	300 batch	500 batch	100 batch	300 batch	500 batch
Kucing Hitam	0.0095	0.0016	0.00067	0.0054	0.00039	0.085
Kucing Putih	0.0087	0.0060	0.0030	0.012	0.00018	0.00021

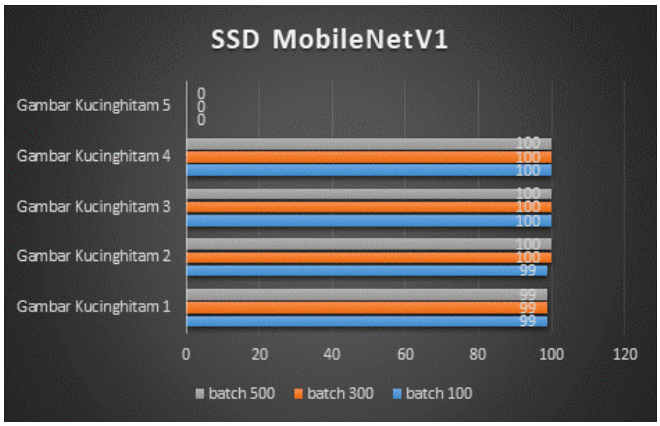
Tabel 2. Rata-rata *Train Loss* kedua Arsitektur

Label	SSD MobileNetV1	SSD ResNet
Jenis	Rata-rata	Rata-rata
Kucing Hitam	0.003923	0.030263
Kucing Putih	0.0059	0.00413

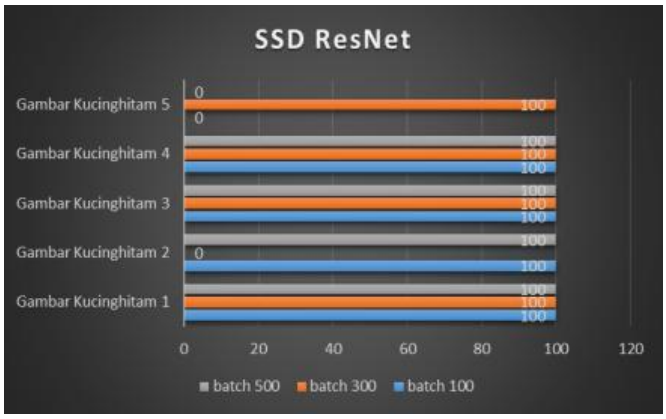
G. Evaluasi model Akurasi

SSD MobileNetV1 memiliki akurasi rata-rata 99,66666667% dengan kesalahan tiga gambar dikarenakan memiliki dua *bounding box*, *ResNet* Akurasi rata-rata 100% dengan kesalahan tiga

gambar dikarenakan memiliki dua *bounding box* dengan objek hewan Kucing Hitam.

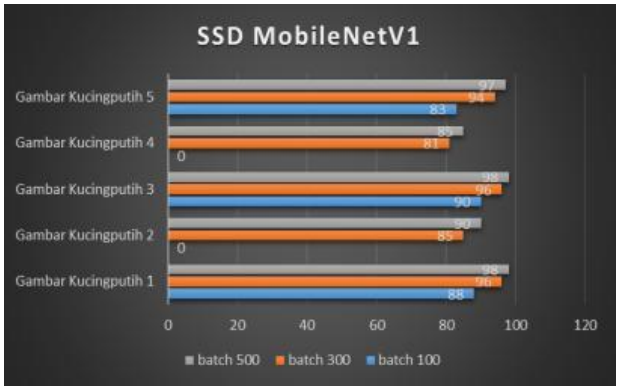


Gambar 11. Perbandingan *SSD MobileNetV1* kucinghitam

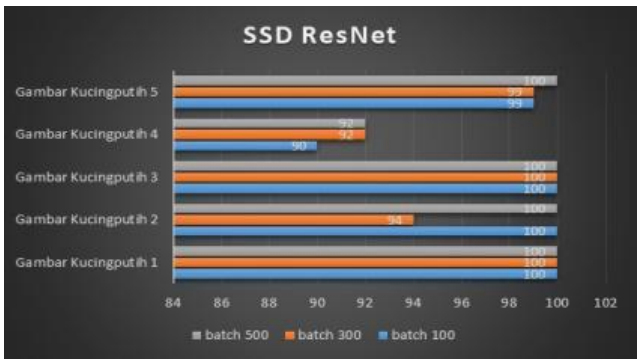


Gambar 12. Perbandingan *SSD ResNet* kucinghitam

MobileNetV1 akurasi rata-rata 78,733% dengan dua kesalahan gambar dikarenakan mesin tidak dapat mendeteksi objek hewan Kucing Putih, *SSD ResNet* Rata-rata Akurasi 97,8% dengan tidak ada kesalahan gambar dengan objek hewan Kucing Putih.



Gambar 13. Perbandingan *SSD MobileNetV1* kucingputih



Gambar 14. Perbandingan *SSD ResNet* kucingputih

Kesimpulan dan Saran

Didalam penelitian ini peneliti mempunyai beberapa kesimpulan, yaitu:

1. Kualitas pengambilan gambar dan proses anotasi memberikan label berpengaruh besar terhadap hasil dari akurasi model yang akan dilatih.

2. Perbedaan jumlah *training model* dalam *batch* berpengaruh terhadap baiknya model yang akan diuji.

3. *SSD MobileNetV1* memiliki *train loss* lebih besar dengan nilai rata-rata 0.003923 pada Kucing Hitam dan nilai rata-rata 0.0059 Kucing Putih jika dibandingkan dengan *SSD ResNet* dengan nilai rata-rata 0.030263 pada Kucing Hitam dan nilai rata-rata 0.00413 pada Kucing Putih.

4. *SSD ResNet* memiliki akurasi lebih tinggi dengan nilai rata-rata 100% pada Kucing Hitam dan 97.9% pada Kucing Putih jika dibandingkan dengan *SSD MobileNetV1* dengan nilai rata-rata 99,66666667% pada Kucing Hitam dan nilai rata-rata 78,733% pada Kucing Putih.

Didalam penelitian ini peneliti mempunyai beberapa saran, yaitu :

1. Gunakan *dataset* yang baik untuk menghasilkan akurasi yang baik.

2. Komparasi menggunakan Arsitektur *Convolutional Neural Network* lain.

Referensi

[1] E. Wijaya, "Analisis Penggunaan Algoritma Breadth First Search Dalam Konsep Artificial Intellegencia," *Time*, vol. II, no. 2, pp. 18–26, 2013.

[2] A. P. Mahardika, "Sistem Pakar Mendeteksi Penyakit Dalam dengan Metode Backward Chaining Menggunakan Visual Basic 2010," pp. 23–34, 2010.

[3] F. D. Wihartiko *et al.*, "Blockchain Dan Kecerdasan Buatan Dalam Pertanian : Blockchain and Artificial Intelligence in Agriculture :," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 8, no. 1, pp. 177–188, 2021, doi: 10.25126/jtiik.202184059.

[4] J. Deng, X. Li, and Y. Fang, "Few-shot Object Detection on Remote Sensing Images," *arXiv*, pp. 1–12, 2020.

[5] "Wibowo - 2017 - 10 Fold-Cross Validation." .

[6] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual Networks of Residual Networks: Multilevel Residual Networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 6, pp. 1303–1314, 2018, doi: 10.1109/TCSVT.2017.2654543.

[7] P. R. Aningtiyas, A. Sumin, and S. Wirawan, "Pembuatan Aplikasi Deteksi Objek Menggunakan TensorFlow Object Detection API dengan Memanfaatkan SSD MobileNet V2 Sebagai Model Pra - Terlatih," *J. Ilm. Komputasi*, vol. 19, no. 3, pp. 421–430, 2020, doi: 10.32409/jikstik.19.3.68.

[8] F. E. Ramadhan, "Penerapan Image Classification Dengan Pre-Trained Model Mobilenet Dalam Client-Side Machine Learning," 2020.

[9] F. G. Mohammadi, M. H. Amini, and H. R. Arabnia, "An Introduction to Advanced Machine Learning : Meta Learning Algorithms, Applications and Promises," *Adv. Intell. Syst. Comput.*, vol. 1123, pp. 129–144, 2020, doi: 10.1007/978-3-030-34094-0_6.

[10] "Garbade - 2018 - Understanding few-shot learning in machine learning." .

[11] P. Chyan, "Penerapan Image Enhancement Algorithm Untuk Meningkatkan Kualitas Citra Tak Bergerak," *Maj. Ilm. INTI*, vol. 12, no. 2, pp. 278–281, 2017.

[12] "Learners - 2019 - Mengenal Machine Learning." .

[13] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-Learning in Neural Networks : A Survey," pp. 1–20, 2020.

[14] E. Retnoningsih and R. Pramudita, "Mengenal Machine Learning Dengan Teknik Supervised dan Unsupervised Learning Menggunakan Python," vol. 7, no. 2, pp. 156–165, 2020.

[15] M. A. Pangestu and H. Bunyamin, "Analisis Performa dan Pengembangan Sistem Deteksi Ras Anjing pada Gambar dengan Menggunakan Pre-Trained CNN Model," *J. Tek. Inform. dan Sist. Inf.*, vol. 4, no. 2, pp. 337–344, 2018.

[16] Reyvan, "Belajar Python dengan Google Colab,"2021.https://www.dqlab.id/belajar-python-dengan-google-colab.

[17] K. He, "Deep Residual Learning for Image Recognition."

[18] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *arXiv*, vol. 1, no. 1, pp. 1–34, 2019.

[19] T. Lin, C. L. Zitnick, and P. Doll, "Microsoft COCO : Common Objects in Context," pp. 1–15.

[20] "Kucing dan Jenisnya (155 ras)," 2020. http://www.kucing.biz/_kucing.php?_i=1&jenis=Domestic-Shorthaired.