Penerapan Algoritma Convolutional Neural Network dalam Klasifikasi Telur Ayam Fertil dan Infertil Berdasarkan Hasil Candling

Article *in* Jurnal Informatika Universitas Pamulang · January 2020

DOI: 10.32493/informatika.v5i4.8556

1 author:

Muhammad Rizky Firdaus
Trilogi University
5 PUBLICATIONS **3** CITATIONS

SEE PROFILE

Jurnal Informatika Universitas Pamulang **ISSN: 2541-1004**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2622-4615**
Vol. 5, No. 4, Desember 2020 (563-571) **10.32493/informatika.v5i4.8556**

Penerapan Algoritma Convolutional Neural Network dalam Klasifikasi Telur Ayam Fertil dan Infertil Berdasarkan Hasil Candling

Muhammad Rizky Firdaus

Fakultas Industri Kreatif dan Telematika, Universitas Trilogi, Jl. Duren Tiga Timur No.30 Kota Jakarta Selatan, Indonesia, 12760

e-mail: rizkyfirdaus0309@gmail.com

Submitted Date: December 25th, 2020 Reviewed Date: December 31th, 2020
Revised Date: December 31th, 2020 Accepted Date: January 05th, 2021

Abstract

Fertile chicken eggs are eggs that can hatch because these eggs have a development in the form of dots of blood and blood vessels or can be called an embryo, while infertile chicken eggs are a type of egg that cannot be hatched because there is no embryo development in the hatching process. Inspection of infertile chicken eggs must be carried out especially for breeders who will carry out the selection and transfer of fertile chicken eggs and infertile chicken eggs. However, currently, the selection of fertile and infertile chicken eggs is still using a less effective way, namely only by looking at the egg shell or called candling, this process is certainly less accurate to classify which eggs are fertile and infertile eggs because not all breeders are able to see the results of the eggs properly. candling so that the possibility of prediction errors. Therefore, in this study, a classification of fertile chicken eggs and infertile chicken eggs will be carried out based on candling results using the Convolutional Neural Network method. From the results of the classification carried out, the percentage of accuracy obtained for the classification of fertile and infertile chicken eggs is 98% and an error of 5%.

Keywords: Fertile Eggs; Infertile Eggs; Convolutional Neural Network; Deep Learning

Abstrak

Telur fertil merupakan telur yang dapat menetas karena pada telur ini terdapat suatu perkembangan yang berbentuk nokta darah atau bisa disebut sebagai embrio sedangkan Telur ayam infertil merupakan jenis telur yang tidak dapat ditetaskan karena tidak adanya perkembangan embrio pada proses penetasan. Pemeriksaan telur ayam infertil harus dilakukan terutama bagi para peternak yang akan melakukan penseleksian serta pemindahan mana telur ayam fertil dan telur ayam infertil. Namun, saat ini penseleksian telur ayam fertil dan infertil ini masih menggunakan cara yang kurang efektif yaitu hanya dengan meneropong cangkang telur atau disebut dengan candling proses ini tentunya kurang akurat untuk mengklasifikasikan mana telur fertil dan telur infertil karena tidak semua peternak mampu melihat dengan baik hasil dari candling tersebut sehingga kemungkinan terjadinya kesalahan prediksi. Dari permasalahan tersebut, maka dalam penelitian ini akan dilakukan klasifikasi telur ayam fertil dan telur ayam infertil berdasarkan hasil candling dengan metode Convolutional Neural Network. Dari hasil klasifikasi yang dilakukan, persentase akurasi yang didapatkan untuk klasifikasi telur ayam fertil dan infertil adalah 98% dan error sebesar 5%.

Kata Kunci: Telur Fertil; Telur Infertil; Convolutional Neural Network; Deep Learning

1. Pendahuluan

Salah satu sumber makanan yang sering dikonsumsi di Indonesia yaitu telur. Karena nilai gizi yang dikandung oleh telur ini sangat bermanfaat dan harga yang relatif lebih murah membuat banyak masyarakat Indonesia menyukai sumber makanan ini. Telur ayam adalah salah satu telur yang sering dikonsumsi oleh beberapa masyarakat Indonesia. Bahkan, hampir seluruh kalangan mengkonsumsi telur jenis ini karena

relatif lebih murah dan mudah untuk diperoleh selain itu juga telur ini memiliki nilai gizi yang baik (Maimunah & Rokhman, 2018). Jenis penghasilan telur ini dibagi menjadi dua kelompok yaitu telur yang dapat ditetaskan (fertile) dan telur yang tidak dapat ditetaskan (infertile). Telur fertil merupakan telur yang dapat menetas karena pada telur ini terdapat suatu perkembangan yang berbentuk nokta darah atau bisa disebut sebagai embrio (Nurdiyah & Muwakhid, 2016). Telur fertil ini dapat menetas karena di dalam telur tersebut memiliki sperma dari ayam jantan sehingga telur-telur fertil ini bisa untuk dikonsumsi maupun ditetaskan. Jika telur fertil ini dierami dengan suhu yang tepat maka kemungkinan besar telur ini akan menetas. Namun, tidak semua jenis telur fertil ini dapat menetas karena untuk proses penetasan ini telur harus melalui proses inkubasi selama beberapa jam lalu disimpan pada suhu yang tepat agar embrio berkembang dan kemudian menetas. Telur fertil biasanya akan menetas selama kurang lebih 21 hari. Agar terhindar dari berbagai bakteri dan penyakit pada telur fertil ini, penetasan telur membutuhkan benih berkualitas agar ayam yang menetas dapat sehat dan tahan terhadap penyakit (Diantoro & Santoso, 2017). Sedangkan telur infertil merupakan jenis telur yang tidak dapat ditetaskan karena tidak adanya perkembangan embrio pada proses penetasan. Pada Dasarnya telur ini juga dibuahi oleh jantan namun ketika proses penetasan, telur jenis ini ternyata tidak dapat menetas (ISNAWATI, 2018). Meskipun telur fertil dan telur infertil dapat dikonsumsi, hasil telur dari kedua proses ini terdapat beberapa perbedaan yaitu; menurut penelitian (Nawawi, Rahmad, & Syahputra, 2015) dijelaskan bahwa telur infertil bisa menjadi tempat berkembang biak jamur hal tersebut disebabkan karena terdapat perbedaan suhu pada telur dan suhu yang diwakili oleh termometer inkubator, telur infertil biasanya akan lebih cepat membusuk daripada telur fertil, karena kontaminasi jamur atau bakteri yang ada pada telur infertil dapat menghasilkan tekanan yang memungkinkan telur tersebut pecah di inkubator. Pemeriksaan telur ayam infertil harus dilakukan terutama bagi para peternak yang akan melakukan penseleksian serta pemindahan mana telur ayam fertil dan telur ayam infertil. Namun, saat ini penseleksian telur ayam fertil dan infertil ini masih menggunakan cara yang kurang efektif yaitu hanya dengan meneropong cangkang telur atau disebut dengan *candling* proses ini tentunya kurang akurat untuk mengklasifikasikan mana telur ayam fertil dan telur ayam infertil karena tidak semua peternak mampu melihat dengan baik hasil dari *candling* tersebut sehingga kemungkinan terjadinya kesalahan prediksi. Dengan menerapkan Algoritma *Convolutiunal Neural Network* penseleksian tersebut dapat dilakukan secara lebih efektif sehingga membuat para peternak telur ayam lebih mudah untuk melakukan penseleksian. Penseleksian dilakukan dengan cara meneropong cangkang telur kemudian mengolah hasil peneropongan tersebut menggunakan Algoritma *Convolutional Neural Network* dengan beberapa dataset yang sudah dikumpulkan melalui studi lapangan metode ini mampu untuk mengklasifikan telur ayam fertil dan telur ayam infertil.

2. Tinjauan Pustaka

Dalam penelitian dibutuhkan dukungan penelitian dari penelitian sebelumnya. Penulis telah mengumpulkan tiga literatur yang berhubungan dengan penelitian ini. Penelitian (Arini, Ubaidillah, Wibisono, & Ulum, 2020). Dalam penelitian ini dijelaskan bahwa untuk meningkatkan persentase dalam proses penetasan telur dibutuhkan suatu penanganan yang dapat memisahkan mana telur fertil dan mana telur infertil. Penanganan yang dilakukan untuk pemisahan telur tersebut dapat dilakukan dengan proses *candling* atau peneropongan telur. Hasil dari penelitian ini adalah sebuah sistem yang dapat mengidentifikasi telur yang fertil dan telur infertil. Metode yang digunakan dalam penelitian ini adalah *Image Processing*. Terdapat perbedaan dengan penelitian yang dilakukan Nur Farida Arini, Achmad Ubaidillah, Kunto Aji Wibisono dan Miftachul Ulum dengan penelitian yang dilakukan saat ini yaitu metode yang digunakan peneliti saat ini adalah Algoritma *Convolutional Neural Network* meskipun metode ini sama dengan *Image Processing* namun arsitektur yang diterapkan berbeda dan juga pada penelitian saat ini belum ada sistem yang dapat digunakan untuk melakukan klasifikasi telur fertil dan telur infertil. Penelitian (Prasmatio, Rahmat, & Yuniar, 2020). Dalam penelitian ini dijelaskan bahwa jumlah pakar ikan yang ada di Indonesia tidak mencukupi untuk melakukan identifikasi ikan yang ada Indonesia. Oleh sebab itu dibutuhkan suatu sistem yang dapat membantu para pakar tersebut dalam mengidentifikasi jenis ikan. Sistem identifikasi ini akan menerapkan Algoritma *Convolutional Neural Network* dalam proses identifikasinya. Hasil dari penelitian ini adalah suatu sistem yang dapat mempermudah para pakar ikan dalam mengidentifikasi jenis-jenis ikan yang ada di

Indonesia. Perbedaan penelitian yang dilakukan R. Mehindra Prasmatio, Basuki Rahmat, Intan Yuniar dengan penelitian saat ini adalah penerapan dari metode yang dilakukan, digunakan untuk mengidentifikasi jenis-jenis ikan yang ada di Indonsia dan pada penelitian saat ini metode diterapkan dalam mengklasifikasi jenis telur fertil dan telur infertil. Penelitian (Nurdiyah, Santosa, & Pramunendar, 2015). Dalam penelitian dijelaskan bahwa pengujian fertilitas telur masih banyak yang menggunakan cara konvensional yaitu dengan memanfaatkan proses *candling* atau peneropongan, sehingga membuat waktu yang lama untuk pemrosesan tersebut. Dari permasalahan tersebut penelitian ini memberikan solusi dengan menggunakan teknologi *Computer Vision* yang dapat mengenali objek berdasarkan ciri-ciri dari objek tersebut sehingga dapat diklasifikasikan. Hasil dari penelitian ini adalah sebuah teknologi yang dapat mempermudah untuk melakukan pengujian fertilitas telur dengan menerapkan metode *Support Vector Machine*. Perbedaan penelitian yang dilakukan dengan penelitian saat ini adalah metode yang diterapkan dalam klasifikasi telur fertil dan infertil. Pada penelitian Dewi Nurdiyah, Stefanus Santosa, Ricardus Anggi Pramunendar menerapkan *Support Vector Machine* sedangan penelitian saat ini menggunakan *Convolutional Neural Network*.

3. Metodologi Penelitian

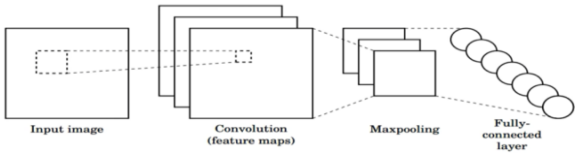
Metode penelitian yang dilakukan yaitu dengan mengumpulkan beberapa sumber literatur dari penelitian terkait melalui jurnal. Selanjutnya adalah pengambilan sampel data dari telur fertil dan infertil. Lalu mengaplikasikan metode Algoritma *Convolutional Neural Network* untuk mengklasifikasikan telur fertil dan telur infertil.

A. Pengumpulan Data Telur

Pengumpulan data telur dilakukan dengan pengambilan gambar hasil penoropongan telur di lapangan. Data-data yang telah didapatkan tersebut nantinya akan digunakan untuk proses klasifikasi menggunakan Algoritma *Convolutional Neural Network*.

B. Algoritma *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah algoritma yang termasuk dalam jenis *Deep Neural Network*, karena algoritma ini memiliki kedalaman jaringan yang tinggi. Algoritma CNN juga merupakan pengembangan dari *Multilayer Perceptron* yang dirancang untuk pengolahan data berbentuk dua dimensi.

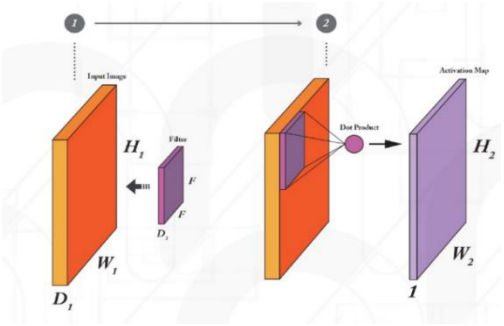


Gambar 1. Proses *Convolutional Neural Nework*

Algoritma Covolutional Neural Network lebih banyak diimplementasikan pada pengolahan data citra (Putra, 2016). Dalam Algoritma Convolutional Neural Network setiap *neuron* dipresentasikan dalam bentuk dua dimensi, cara kerja tersebut hampir mirip pada Multilayer Perceptron perbedaannya yaitu pada Multilayer Perceptron neuron dipresentasikan dalam bentuk ukuran satu dimensi. Algoritma Convolutional Neural Network mempunyai beberapa layer yang mana layer tersebut nantinya digunakan untuk melakukan filtering pada setiap proses komputasinya, proses tersebut dinamakan *training*. Algoritma CNN ini terdiri dari tiga jenis *layer* yaitu; *Convolutional Layer*, *Pooling Layer* dan *Fully Connected Layer*, ketika *layer* ini telah ditumpuk menjadi satu arsitektur, Algoritma CNN telah terbentuk (O'Shea & Nash, 2015). Proses dari tiga arsitektur tersebut dapat dilihat pada gambar 1.

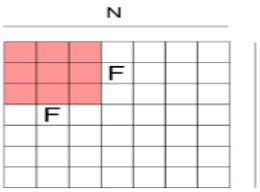
1) *Convolutional Layer*

Convolutional layer melakukan operasi konvolusi, yaitu mengubah *input* menjadi *feature maps* dengan melakukan operasi *dot* antara matriks input dengan filter (Wulandari, Yasin, & Widiharih, 2020). Filter ini memiliki ukuran *height* (panjang), *width* (lebar), *heavy* (tebal) tertentu. Berikut adalah gambaran dari proses *Convolutional Layer*.



Gambar 2. *Convolutional Layer*

Setiap posisi gambar akan menghasilkan sebuah nilai yang merupakan perkalian titik antara bagian gambar dengan filter yang digunakan. Berikut adalah ilutrasi proses *convolutional layer*.



Gambar 3. Proses *Convolutional Layer*

Ukuran spasial dari proses output gambar di atas, dihitung menggunakan rumus:

$$\frac{N - F + 2P}{S} + 1$$

Keterangan:
N: Ukuran spasial (tinggi H1 = lebar W1) dari input yang ada pada gambar.
F: Merupakan ukuran spasial filter
P: Suatu parameter di mana parameter ini menentukan jumlah pixel, lalu akan ditambahkan pada setiap sisi dari input, disebut juga sebagai Zero Padding
S: Besaran pergeseran filter pada setiap proses komputasi disebut juga dengan *Stride.*

Jika dilihat dari ilustrasi gambar di atas maka perhitungan komputasinya adalah sebagai berikut.

- N = 7
- F = 3
- P (Padding) = 0
- S (Stride) = 2 (Didapatkan berdasarkan pergeseran filter)

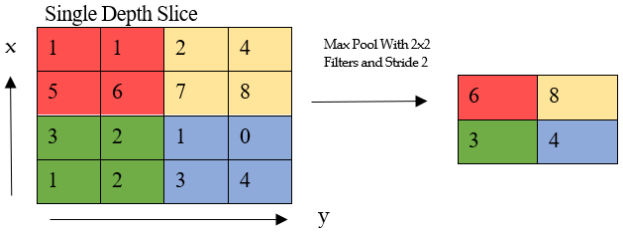
Stride 1 -> $\frac{7-3+0}{1} + 1 = 5$

Stride 2-> $\frac{7-3+0}{1} + 2 = 3$

Proses komputasi ini akan diulang beberapa kali di setiap filter yang berbeda sampai menghasilkan beberapa kumpulan dari *activation maps* yaitu sebuah gambar baru.

2) *Pooling Layer (Max Pooling)*

Merupakan proses setelah *Convulutional Layer*. *Pooling layer* akan membagi beberapa nilai output dari proses *convolution layer* menjadi matriks yang lebih kecil, dengan mengambil nilai maksimal dari nilai matriks sebelumnya. Tujuannya adalah untuk mengurangi dimensi fitur matriks (Felix, Wijaya, Sutra, Kosasih, & Sirait, 2020). Proses *Pooling Layer* dapat dilihat pada Gambar 4.



Gambar 4. Proses *Pooling Layer*

3) *Fully Connected Layer*

Hasil *feature* map dari kedua *layer* sebelumnya masih berbentuk multidimensional array oleh sebab itu diperlukan proses *reshape feature map* agar menjadi sebuah *vector*. Merupakan *layer* yang digunakan untuk melakukan transformasi pada data dimensi dengan tujuan data tersebut dapat diklasifikasikan secara linear. Neuron-neuron yang ada pada *convolutional layer* harus ditransformasikan kembali menjadi data *array* satu dimensi setelah itu dimasukkan ke dalam arsitektur ini. Proses ini bertujuan agar data yang ada tidak kehilangan informasi spasial dan tidak *reversible*. Arsitektur ini harus

diimplementasikan pada akhir jaringan. (Ilahiyah & Nilogiri, 2018).

4. Hasil dan Pembahasan

A. Data Telur

Data telur yang telah dikumpulkan dibagi menjadi dua kelas yaitu telur fertil dan telur infertil. Data kelas pertama yaitu telur fertil, data ini terkumpul menjadi 111 data dan data kelas kedua yaitu telur infertil terkumpul menjadi 138 data. Setelah itu kedua kelas ini dipecah kembali menjadi data *training* atau data latih dan data validasi. Data *training* telur fertil dibagi menjadi 88 data dan data *training* telur infertil dibagi menjadi 110 data. Lalu data validasi untuk telur fertil dibagi menjadi 23 data dan data validasi telur infertil dibagi menjadi 28 data. Data-data berikut dapat dilihat pada tabel 1.

Tabel 1. Pembagian Data Telur Fertil dan Infertil

Jenis Data	Telur Fertil	Telur Infertil
Data Asli	111 Data	138 Data
Data Training	88 Data	110 Data
Data Validasi	23 Data	28 Data

B. Penerapan *Convolutional Neural Network*

Proses penulisan *code* menggunakan *Platform* bernama *Google Colab* di mana *platform* ini dikhususkan untuk keperluan pengembangan dari *Machine Learning*. Sebelum memasuki proses penerapan *Convolutional Neural Network* pada klasifikasi telur fertil dan infertil ada beberapa *library* yang digunakan untuk proses tersebut yaitu *library Tensorflow* dan *Keras*. Tensorflow atau biasa disingkat dengan TF merupakan suatu *library open-source* yang digunakan untuk pengembanan *Machine Learning*. *TensorFlow* akan membagi sebuah komputasi graf yang telah ditentukan sebelumnya lalu menjalankannya secara paralel. Dengan *Tensorflow* ini, proses pelatihan model dari jaringan saraf akan menjadi lebih efisien. Lalu *library Keras*, *library* ini merupalan *library* yang dibangun di atas *backend TensorFlow* yang berfungsi untuk membuat sebuah *image classifier*. Dengan library Keras proses membangun jaringan saraf tiruan untuk klasifikasi gambar akan menjadi lebih mudah. Penggunaan dari *library* tersebut dilakukan dengan cara mengimpor pada *header*. Pada Gambar 5 dapat dilihat baris kode untuk mengimpor *library* tersebut.

```
import tensorflow as tensorf
import sys
import zipfile as zipper
import os
import shutil
import numpy as nmp
import matplotlib.pyplot as pyp
import matplotlib.image as matpimg
from google.colab import files
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import Sequence
from tensorflow.keras.optimizers import RMSprop
from sklearn.model_selection import train_test_split
%matplotlib inline
```

Gambar 5. Mengimpor *library Tensorflow* dan *Keras*

Selanjutnya adalah membuat direktori untuk menampung data latih dan data validasi dari *dataset* telur fertil dan infertil. Setelah membuat dua direktori data latih dan data validasi selanjutnya *dataset* telur fertil dan infertil dipecah kembali ke dalam direktori latih dan direktori validasi. Proses pembagian data dapat dilihat pada Gambar 6.

```
[7] base_dir = '/content/Data' #base direktori untuk data telur
    train_dir = os.path.join(base_dir , 'train') #membuat nama direktori data train
    val_dir = os.path.join(base_dir, 'val') #membuat nama direktori data validasi

    #Membuat data direktori train dan validasi
    os.mkdir(train_dir)
    os.mkdir(val_dir)

[8] fertil_dir = os.path.join(base_dir, 'Fertil') #inisialisasi data telur fertil
    infertil_dir = os.path.join(base_dir, 'Infertil') #inisialisasi data telur infertil

    # Memecah data telur fertil menjadi data train dan validasi
    train_fertil, val_fertil = train_test_split(os.listdir(fertil_dir), test_size= 0.2)

    #Memecah data telur infertil menjadi data train dan validasi
    train_infertil, val_infertil = train_test_split(os.listdir(infertil_dir), test_size = 0.2)

[10] train_fertil_dir = os.path.join(train_dir, 'Fertil')
     train_infertil_dir = os.path.join(train_dir, 'Infertil')

     val_fertil_dir = os.path.join(val_dir, 'Fertil')
     val_infertil_dir = os.path.join(val_dir, 'Infertil')

     #Membuat direktori baru di dalam directory train dan val
     if not os.path.exists(train_fertil_dir):
       os.mkdir(train_fertil_dir)
     if not os.path.exists(train_infertil_dir):
       os.mkdir(train_infertil_dir)

     if not os.path.exists(val_fertil_dir):
       os.mkdir(val_fertil_dir)
     if not os.path.exists(val_infertil_dir):
       os.mkdir(val_infertil_dir)

#Menyalin data train dan data val ke dalam directory baru
import shutil

#train data
for i in train_fertil:
  shutil.copy(os.path.join(fertil_dir, i), os.path.join(train_fertil_dir, i))
for i in train_infertil:
  shutil.copy(os.path.join(infertil_dir, i), os.path.join(train_infertil_dir, i))


#val data
for i in val_fertil:
  shutil.copy(os.path.join(fertil_dir, i), os.path.join(val_fertil_dir, i))
for i in val_infertil:
  shutil.copy(os.path.join(infertil_dir, i), os.path.join(val_infertil_dir, i))
```

Gambar 6. Proses Pembagian Data Latih dan Validasi

Setelah proses pembagian data telah selesai, maka selanjutnya adalah mempersiapkan data latih dan data *testing* yang nantinya akan diberikan ke model. Proses ini menggunakana *Image Data Generator* yang telah tersedia di dalam *library Tensorflow*. Fungsi dari *Image Data Generator* adalah melakukan *preprocessing data*, pelabelan sampel dan augmentasi gambar. Proses augmentasi gambar bertujuan untuk membuat data-data baru dari data yang telah ada. Pada gambar 7 dapat dilihat proses dari augmentasi gambar pada setiap *dataset*.

```
#membuat data image generator
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    horizontal_flip = True,
    shear_range = 0.2,
    fill_mode = 'nearest'
)

test_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    horizontal_flip = True,
    shear_range = 0.2,
    fill_mode = 'nearest'
)
```

Gambar 7. Proses Augmentasi Gambar

Selanjutnya adalah melakukan *training* pada model dengan mempersiapkan data latih yang telah dipecah sebelumnya. Proses *training* dapat dilihat pada gambar 8.

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size = (64,64),
    batch_size = 4,
    class_mode = 'binary'
)

validation_generator = test_datagen.flow_from_directory(
    val_dir,
    target_size = (64,64),
    batch_size = 4,
    class_mode = 'binary'
)
```

```
Found 238 images belonging to 2 classes.
Found 91 images belonging to 2 classes.
```

Gambar 8. Proses *training*

Dapat dilihat bahwa *output* menampilkan banyak data dari data latih dan data validasi. Di mana data latih menampilkan 198 gambar yang terbagi menjadi 2 kelas dan data validasi menampilkan 51 gambar yang terbagi menjadi 2 kelas. Setelah proses ini selanjutnya adalah membangun arsitektur dari *Convolutional Neural Network*. Baris kode untuk membangun arsitektur *Convolutional Neural Network* dapat dilihat pada gambar 9.

```
[108] #membuat model menggunakan tensor flow
      model = tensorf.keras.models.Sequential([
              tensorf.keras.layers.Conv2D(32, (3,3),  activation = 'relu',input_shape= (64,64,3)),
              tensorf.keras.layers.MaxPooling2D(2,2),
              tensorf.keras.layers.Conv2D(64,(3,3), activation= 'relu'),
              tensorf.keras.layers.MaxPooling2D(2,2),
              tensorf.keras.layers.Conv2D(64,(3,3), activation= 'relu'),
              tensorf.keras.layers.MaxPooling2D(2,2),
              tensorf.keras.layers.Conv2D(128, (3,3), activation = 'relu'),
              tensorf.keras.layers.MaxPooling2D(2,2),
              tensorf.keras.layers.Dropout(0.5),
              tensorf.keras.layers.Flatten(),
              tensorf.keras.layers.Dense(64, activation= 'relu'),
              tensorf.keras.layers.Dropout(0.5),
              tensorf.keras.layers.Dense(128, activation= 'relu'),
              tensorf.keras.layers.Dropout(0.5),
              tensorf.keras.layers.Dense(256, activation= 'relu'),
              tensorf.keras.layers.Dropout(0.5),
              tensorf.keras.layers.Dense(512, activation= 'relu'),
              tensorf.keras.layers.Dropout(0.5),
              tensorf.keras.layers.Dense(1, activation= 'sigmoid')

      ])
```

Gambar 9. Arsitektur *Convolutional Neural Network*

1. *Convolutional Layer*

Dari baris kode pada gambar 9 proses *Convolutional Layer* terdapat pada fungsi *Conv2D* yang berguna untuk melakukan konvolusi pada ukuran gambar, di mana parameter pertama mendefinisikan jumlah filter, parameter kedua adalah dimensi filter, parameter ketiga adalah fungsi aktivasi di mana fungsi aktivasinya menggunakan *ReLu*, lalu pada proses *Convolutional Layer* pertama terdapat *input shape* yang artinya bentuk *input* dari gambar bertipe RGB dengan ukuran 64x64 *pixel* dengan jumlah *channel* adalah 3. resolusi gambar dari data *training*. Gambar-gambar yang terdapat pada *dataset* memiliki resolusi gambar 150x150 *pixel* dan jumlah *channel* adalah 3 lalu jumlah filter adalah 3x3.

2. *Max Pooling*

Max Pooling terdapat setelah proses konvolusi layer. Proses *Max Poolingi* terdapat pada fungsi *MaxPooling* yang memiliki 1 parameter yaitu jumlah *pooling* dalam bentuk matriks di mana dalam kasus ini jumlah *polling* yang diinisialisasi adalah 2x2 yang berguna sebagai *pixel loss* dan *precise region*. Tujuan dari proses ini adalah untuk mereduksi resolusi gambar sehingga proses pelatihan dari model akan lebih cepat. Proses konvolusi layer dan *max pooling* dilakukan beberapa kali, dalam kasus ini 2 proses tersebut dilakukan sebanyak

4 kali dengan nilai *layer* yang berbeda. *Input shape* hanya dilakukan pada proses konvolusi di *layer* pertama yaitu 32.

3. *Fully Connected Layer*

Setelah dua proses tadi, hasil dari matriks *input* masih berbentuk 2 dimensi sehingga perlu dilakukan perubahan kembali menjadi matriks 1 dimensi. Proses perubahan ini menggunakan fungsi *Flatten*, di mana fungsi ini berguna untuk meratakan hasil dari *feature map input* yang telah diproses pada 2 layer sebelumnya. Lalu juga ada fungsi *Dense* yang berguna untuk menambahkan lagi layer pada proses *fully Connected* dengan parameter *units* dan aktifasi yang digunakan. *Units* menandakan jumlah *Node* yang ada pada *hidden layer* dan nilai yang diambil adalah nilai jumlah antara *Node Input* dan *Node Ouput*. Selanjutnya adalah menginisialisasi *output layer* dengan menggunakan fungsi *Dense* dengan parameter pertama adalah jumlah *node* yang menandakan label kelas dari data, karena pada kasus ini hanya terdapat 2 kelas maka jumlah unit diberikan nilai 1 yang artinya 0 adalah kelas pertama dan 1 adalah kelas kedua dan aktifasi yang digunakan adalah *Sigmoid* untuk klasifikasi dua kelas. Setelah arsitektur CNN telah dibangun, selanjutnya adalah melakukan kompilasi dari model tersebut. Pada gambar 10 dapat dilihat baris kode yang melakukan proses kompilasi.

```
model.compile(
    loss = 'binary_crossentropy',
    optimizer = tensorf.optimizers.Adam(),
    metrics = ['accuracy']
)
```

Gambar 10. Proses Kompilasi Model

Pada proses kompilasi terdapat 3 parameter yaitu: parameter *loss* yang berfungsi untuk menentukan *loss function* dengan menggunakan *library* dari keras yaitu *binary_crossentropy*, parameter kedua adalah parameter optimasi yang berguna untuk penentuan algoritma *stochastic gradient descent* dengan menggunakan *library Adam* dari *TensorFlow*, lalu parameter ketiga adalah parameter *metrics* yang berfungsi untuk penentuan performa dari *metrics*. Selanjutnya adalah melakukan proses *training* pada model menggunakan metode *fit*. Baris kode untuk proses *training* dapat dilihat pada Gambar 11.

```
history = model.fit(
    train_generator,
    steps_per_epoch = 15,
    epochs = 10,
    validation_data = validation_generator,
    validation_steps = 5,
    verbose =2
)

score = model.evaluate(validation_generator, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Gambar 11. Proses *Training*

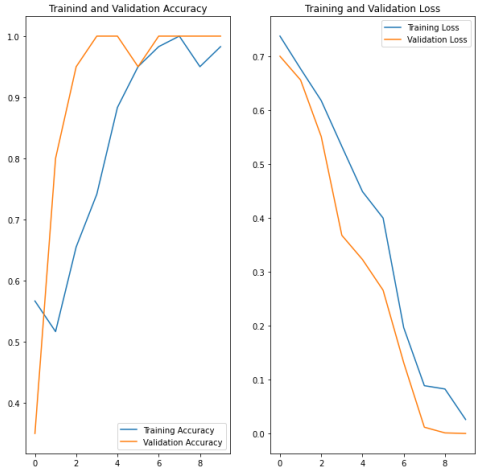
Metode *fit* ini memiliki 6 paramaeter di mana parameter pertama adalah variabel dari proses *Image Data Generator* pada data *training*, parameter kedua adalah jumlah *batch* yang akan dieksekusi pada setiap *epochs*, parameter ketiga adalah jumlah *epochs* yang akan dilakukan, parameter kedua adalah jumlah validasi pada setiap *epochs* dan parameter *verbose* berguna untuk menampilkan *output* dari proses *training*. Jumlah *verbose* yang digunakan adalah 2 yang artinya *output* yang ditampilkan hanya akan menyebutkan jumlah *epochs* tanpa menampilkan animasi proses. Lalu setelah fungsi *fit*, dilakukan evaluasi data *loss* dan *accuracy* dari proses *training* menggunakan fungsi *evaluate*. Hasil dari proses *training* dapat dilihat pada tabel 2.

Tabel 2. Hasil *Training* Model

Jumlah *Epochs*	*Loss*	*Accuracy*	*Val Loss*	*Val Accuracy*
1	0.7374	0.5667	0.7000	0.3500
2	0.6768	0.5167	0.6559	0.8000
3	0.6178	0.6552	0.5510	0.9500
4	0.5328	0.7414	0.3678	1.0000
5	0.4489	0.8833	0.3229	1.0000
6	0.3998	0.9500	0.2659	0.9500
7	0.1964	0.9828	0.1313	1.0000
8	0.0887	1.0000	0.0117	1.0000
9	0.0826	0.9500	0.0011	1.0000
10	0.0260	0.9828	4.6433e-	1.0000
Test Loss	0.05663638934493065			
Test accuracy	0.9803921580314636			

Dapat dilihat pada tabel 2, rata-rata *loss* yang didapat adalah 0.056 jika dihitung dalam persentasi adalah 6% sedangkan jumlah

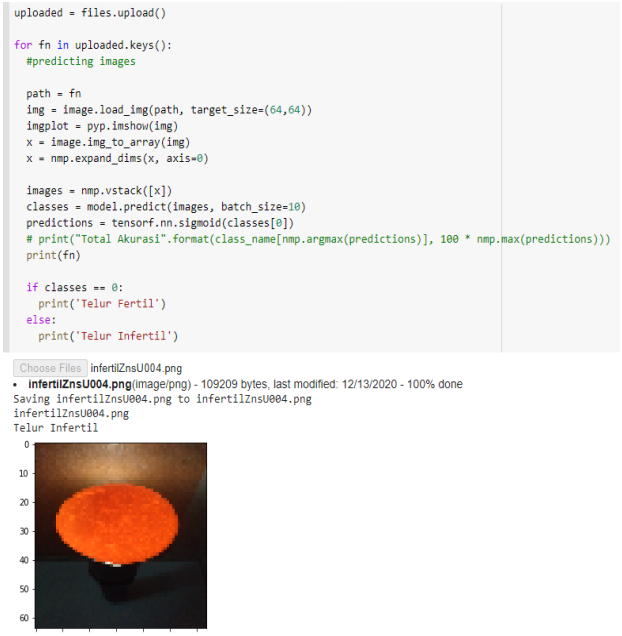
accuracy yang didapatkan adalah 0.98 jika dalam persentasi adalah 98%. Gambar 12 adalah visualisasi grafik proses *training* model.



Gambar 12. Visualisasi Grafik Proses *Training*

Visualisasi grafik pada gambar 12 menunjukkan bahwa proses *training* pada akurasi mengalami kenaikan pada setiap *epochs* dan pada *loss* mengalami penurunan. Selanjutnya pada tahap terakhir adalah melakukan klasifikasi telur ayam fertil dan infertil. Proses klasifikasi dapat dilihat pada gambar 13 untuk telur ayam fertil dan 14 untuk telur ayam infertil.

Gambar 13. Proses Klasifikasi Telur Ayam Fertil



Gambar 14. Proses Klasifikasi Telur Ayam Infertil

Pada gambar 13 dan 14 dapat dilihat bahwa model yang telah dibuat mampu untuk mengklasifikasikan antara telur ayam fertil dan telur ayam infertile.

5. Kesimpulan dan Saran

Dari hasil penelitian ini dapat disimpulkan bahwa:

1. Penerapan Algoritma *Convolutional Neural Network* dapat diterapkan untuk mengklasifikasi telur ayam fertil dan telur ayam infertil
2. Nilai akurasi yang didapatkan pada proses klasifikasi adalah 98% dan nilai *error* sebesar 5%

Sedangkan kekurangan dari penelitian ini adalah:

1. Data telur fertil yang diambil adalah telur yang sudah dierami selama 5 sampai 10 hari sehingga klasifikasi telur hanya dapat dilakukan pada hari-hari tersebut.
2. Dari grafik yang ada menunjukkan bahwa adanya kemungkinan jika proses *training* model mengalami *overfitting* karena nilai antara *training* dan validasi pada beberapa *epochs* terdapat perbedaan yang cukup signifikan. Hal tersebut disebabkan karena dataset yang didapatkan tidak cukup banyak untuk proses pelatihan dan validasi.

Hasil penelitian ini masih dapat dikembangkan kembali dengan memperbanyak dataset agar proses *training* dapat terhindar dari *overfitting*.

Referensi

Arini, N. F., Ubaidillah, A., Wibisono, K. A., & Ulum, M. (2020). Identifikasi embrio dalam telur berbasis image processing. *Jurnal Teknik Elektro Dan Komputasi (ELKOM)*, *2*(1), 11–19.

Diantoro, A., & Santoso, I. B. (2017). Classification of Egg Fertility on the Image of Kampong Chicken Egg Using the Frequency Distribution Feature and Naive Bayes Classifier Algorithm's. *Proceedings of the International Conference on Green Technology*, *8*(1), 446–453.

Felix, F., Wijaya, J., Sutra, S. P., Kosasih, P. W., & Sirait, P. (2020). Implementasi Convolutional Neural Network Untuk Identifikasi Jenis Tanaman Melalui Daun. *Jurnal SIFO Mikroskil*, *21*(1), 1–10.

Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem Dan Teknologi Informasi Indonesia)*, *3*(2), 49–56.

Isnawati, I. (2018). Klasifikasi Citra Candling Telur Ayam Kampung Dengan Metode LVQ.

Maimunah, M., & Rokhman, T. (2018). Klasifikasi Penurunan Kualitas Telur Ayam Ras Berdasarkan Warna Kerabang Menggunakan Support Vector Machine. *INFORMATICS FOR EDUCATORS AND PROFESSIONAL: Journal of Informatics*, *3*(1), 43–52.

Nawawi, M. Z., Rahmad, R., & Syahputra, M. (2015). Klasifikasi telur fertil dan infertil menggunakan jaringan saraf tiruan multilayer perceptron berdasarkan ekstraksi fitur warna dan bentuk. *Jurnal Teknologi Informasi Dan Komunikasi*, *4*(2), 100–109.

Nurdiyah, D., & Muwakhid, I. A. (2016). Perbandingan Support Vector Machine Dan K-Nearest Neighbor Untuk Klasifikasi Telur Fertil dan Infertil Berdasarkan Analisis Texture GLCM. *Jurnal Transformatika*, *13*(2), 29–34.

Nurdiyah, D., Santosa, S., & Pramunendar, R. A. (2015). Klasifikasi Citra Telur Fertil dan Infertil Dengan Analisis Tekstur Gray Level Co-Occurence Matrix dan Support Vector Machine. *Cyberku Journal*, *11*(2), 116–126.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *ArXiv Preprint ArXiv:1511.08458*.

Prasmatio, R. M., Rahmat, B., & Yuniar, I. (2020). Deteksi dan Pengenalan Ikan Menggunakan Algoritma Convolutional Neural Network. *Jurnal Informatika Dan Sistem Informasi (JIFoSI)*, *1*(2), 510–521.

Putra, I. (2016). *Klasifikasi citra menggunakan convolutional neural network (CNN) pada caltech 101*. Institut Teknologi Sepuluh Nopember.

Wulandari, I., Yasin, H., & Widiharih, T. (2020). Klasifikasi CItra Digital Bumbu dan Rempah dengan Algoritma Convolutional Neural Network. *Jurnal Gaussian*, *9*(3), 273–282.