DAFTAR PUSTAKA

Adhikari SP, Meng S, Wu Y, Mao Y, Ye R, Wang Q, Sun C, Sylvia S, Rozelle S, Raat H, and Zhou H. (2020). Epidemiology, Causes, Clinical Manifestation And Diagnosis, Prevention And Control Of Coronavirus Disease (*COVID-19*) During The Early Outbreak Period: A Scoping Review. Infectious Diseases of Poverty, 9(1), 29. Available: https://pubmed.ncbi.nlm.nih.gov/32183901/.

Albertus, A. (2020). Diagnosis Coronavirus Disease 2019 (COVID-19). Diakses 4 Juli 2022 dari alomedika https://www.alomedika.com/penyakit/penyakit-infeksi/coronavirus-disease-2019 covid- 19/diagnosis.

Alim, M. M. F. (2020). Identifikasi Penyakit Tanaman Tomat Menggunakan Algoritma Convolutional Neural Network Dan Pendekatan Transfer Learning. Fakultas Teknik Universitas Negeri Semarang. Diakses 20 Juli 2022 Available: http://lib.unnes.ac.id/42927/1/5302416053%20-20Fathul%20Alim.pdf

Alzubaidi L. (2021). "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," p. 74. Jurnal Big Data. Diakses 15 Jul 2022 : https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8.

Buyut K, Visq D, (2021) "Penerapan transfer learning pada convolutional meural network dalam deteksi covid-19". Lenteradua. Diakses 1 Juli 2022 http://lenteradua.net/jurnal/index.php/jnanaloka/article/view/38

Hafizhan A. (2020). "Membuat klasifikasi gambar menggunakan keras-tensorflow dipython.Medium.Diakses2Agustus2022https://medium.com/@hafizhan.aliady/membuat-klasifikasi-gambar-images-menggunakan-keras-tensorflow-tf-keras-dan-python-53f7ae953cea

Ilyas M, Rehan H, And Nait-ali A. (2020). "Dtection of *Covid-19* From Cheast *X-ray* Images Using Artificial Intelligence: An Early Review," pp. 1-8, diakses 17 Agustus 2022. http://arxiv.org/abs/2004.05436.

Juniati Dwi, Sapata B, (2019). "Klasifikasi Penyakit Paru Berdasarkan Citra X-Ray Thorax Menggunakan Metode Fraktal Box Counting" diangkes 20 November 2022https://ejournal.unesa.ac.id/index.php/mathunesa/article/view/30272/28110

Mawaddah H, Lilis S, Melda S, (2022). “deteksi penyakit covid-19 pada x-ray dengan pendekatan convolution neural network(CNN)”. Jurnal resti rekayasa sistemdanteknologi.Diakses20Juli2022https://jurnal.iaii.or.id/index.php/RESTI/article/view/3373

Majeed T, Rashid R, Ali D, and Asaad A. (2020). “Covid-19 detection using CNN transfer learning from X-ray Images,” medRxiv, p. Diakses 10 Agustus 2022.https://www.medrxiv.org/content/10.1101/2020.05.12.20098954v2.abstract.

Misnadiarly. (2008). Penyakit Infeksi Saluran Napas Phemonia pada Balita, Orang Dewasa, Usia Lanjut. Pustaka Obor Populer, Jakarta.

Mohd. A. (2021). “Implementasi Arsitektur Xception Untuk Klasifikasi Citra Covid-19 Radiography”. prosiding.ikonik. Diakses 11 Agustus 2022. https://prosiding.konik.id/index.php/konik/article/download/88/80/174.

Nikola B. (2020). “The differences between sigmoid and softmax activation functions”. Medium. Diakses 11 Agustus 2022 https://medium.com/arteos-ai/the-differences-between-sigmoid-and-softmax-activation-function-12adee8cf322

Palmer W. J. (2020). “ACR Releases CT and Chest *X-ray* Guidance Amid *COVID-19*Pandemic”.Diakses22Juli2022:https://www.diagnosticimaging.com/view/acr-releases-ct-and-chest-*x-ray*-guidance-amid-*Covid-19*-pandemic.

Peryanto A, Yudhana A, and Umar R. (2020). “Klasifikasi citra menggunakan convolutional neural network dan k fold cross validation,”. Polibatam Diakses 10 Agustus 2022. http://jurnal.polibatam.ac.id/index.php/JAIC.

Pathari S and Rahul U. (2020). “Automatic detection of *COVID-19* and *Pneumonia* from Chest *X-ray* using transfer learning,”diakses 22 Agustus 2022medRxiv, https://www.medrxiv.org/content/10.1101/2020.05.27.20100297v1

Pneumonia, World Health Organization, (2019). Diakses pada 19 Juli 2022, https://www.who.int/news-room/fact-sheets/detail/*Pneumonia*.

Rahimzadeh M and Attar A. (2020). “A modified deep convolutional neural network for detecting *COVID-19* and *Pneumonia* from chest *X-ray* images

based on the concatenation of Xception and ResNet50V2," Inform. Med. Unlocked, vol. 19, p.

Rawat W dan Wang Z, (2017). "Deep Convolutional Neural Networks for Image Classification : A Comprehensive Review," Neural Comput. 29, hal. 2352–2449 diakses 20 Juli 2022 https://pubmed.ncbi.nlm.nih.gov/28599112/.

Sarang N. (2018). "Understanding AUC- ROC kurve" towardsdatascience. Diakses 20 Juli 2022 https://www.codingninjas.com/codestudio/library/mobilenet

Saqib A, Anwar S, Anwar A, Petersson L, Sharma N, and Blumenstein M,(2020). "COVID19 detection from Radiographs: Is Deep Learning able to handle the crisis?",diakses17Juli2022Resti,https://jurnal.iaii.or.id/index.php/RESTI/article/view/3373.

Syaifulloh M. (2021). "deteksi penyakit pneumonia dan covid-19 menggunakan citra x-ray dengan menggunakan cnn googlenet" digilib, diakses 1 Juli 2022 https://digilib.uinsa.ac.id/49030/

Sethy P dan Behera S (2020). "Detection of coronavirus disease (*Covid-19*) based on deepfeatures, preprints," https://doi. org/10.20944/preprints202003, vol. 300, p. v1.

Singh D, Kumar V, Vaishali, and Kaur M (2020). "Classification of *COVID-19* patients from chest CT images using multi-objective differential evolution–based convolutional neural networks," Eur. J. Clin. Microbiol. Infect. Dis., vol. 39, no. 7, pp. 1379–1389, European Journal Of Clinical Microbiology & infectious diakses 1 Agustus 2022 https://doi.org/10.1007/s10096-020-03901-z.

Sofia N. (2020). "Convolutional neural network," medium. Diakses 20 Juli 2022 Available: https://medium.com/@nadhifasofia/1-convolutional-neuralnetwork-convolutional-neural-network-merupakan\-salah-satu-metode-machine-28189e17335b

LAMPIRAN

Berikut ini merupakan Source Code yang digunakan untuk mengimplementasikan arsitektur Xception, InceptionV3, dan MobileNet dalam mendeteksi penyakit Covid-19.

A. Visualisali Data

```
from google.colab import drive
drive.mount('/content/drive')


import numpy as np
import pandas as pd
import os

import matplotlib.pyplot as plt
import tensorflow as tf
import matplotlib.image as mpimg
from tensorflow.keras.preprocessing.image import load_img, img
_to_array
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import load_img, img
_to_array
from tensorflow.keras.preprocessing.image import ImageDataGene
rator
from sklearn.metrics import classification_report, log_loss, a
ccuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
from scipy import interp
from itertools import cycle

directory = '/content/drive/MyDrive/ctscan/dataset_covid'

File=[]
for file in os.listdir(directory):
    File+=[file]
print(File)

# Visualisasi citra asli
directory = "/content/drive/MyDrive/ctscan/dataset_covid/COVID
19/COVID19(100).jpg"
```

```
img = mpimg.imread(directory)
imgplot = plt.imshow(img)
plt.show()

#Visualisasi citra yang diresize
img_resize = load_img(directory, grayscale=False, color_mode='
rgb', target_size=(299,299))
plt.figure(figsize=(3,3))
imgplot = plt.imshow(img_resize)

# Mengubah gambar menjadi array
img_arr = img_to_array(img_resize)

# Menampilkan array
img_arr

# Melakukan normalisasi
img_norm = img_arr/255.0

# Menampilkan array hasil normalisasi
img_norm
```

B. Preprocessing Data

```
# Preprocessing data citra
dataset = []
mapping = {'COVID19':0, 'PNEUMONIA':1, 'NORMAL':2}
count = 0

for file in os.listdir(directory):
    path = os.path.join(directory,file)
    for im in os.listdir(path):
        image = load_img(os.path.join(path,im), grayscale=Fals
e, color_mode='rgb', target_size=(299,299))
        image = img_to_array(image)
        image = image/255.0
        dataset.append([image,count])
    count=count+1

# Mengubah data menjadi array
data,labels0=zip(*dataset)
labels1=to_categorical(labels0)
data=np.array(data)
labels=np.array(labels1)
```

```
print(data.shape)
print(labels.shape)


# Pembagian data training dan testing
train_x,test_x,train_y,test_y=train_test_split(data,labels,tes
t_size=0.2,random_state=13)

ter = np.argmax(test_y, axis=1)

np.unique(ter, return_counts=True)

len(train_x), len(test_x)

# Menampilkan bentuk dimensi data training dan testing
print(train_x.shape)
print(test_x.shape)
print(train_y.shape)
print(test_y.shape)

# menampilkan augmentasi
datagen = ImageDataGenerator(horizontal_flip=True,vertical_fli
p=True,rotation_range=20,zoom_range=0.2,
                     width_shift_range=0.2,height_shift_range=0
.2,shear_range=0.1,fill_mode="nearest")
```

C. Membangun dan melatih arsitektur Xception, InceptionV3, dan MobileNet

1. Arsitektur Xception

```
# Model Xception
pretrained_model3 = tf.keras.applications.Xception(input_shape
=(299,299,3),include_top=False,weights='imagenet',pooling='avg
')
pretrained_model3.trainable = False


# membangun model Xception
inputs3 = pretrained_model3.input
x3 = tf.keras.layers.Dense(128, activation='relu')(pretrained_
model3.output)
outputs3 = tf.keras.layers.Dense(3, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
```

```
model.compile(loss = 'categorical_crossentropy', optimizer='ad
am', metrics=['accuracy'])


model.summary()

# plot the model including the sizes of the model
tf.keras.utils.plot_model(model, show_shapes=True)


# melatih Model
his=model.fit(datagen.flow(trainx,trainy,batch_size=32),valida
tion_data=(testx,testy),epochs=30)
```

2. Arsitektur InceptionV3

```
pretrained_model3 = tf.keras.applications.InceptionV3(input_sh
ape=(299,299,3),include_top=False,weights='imagenet',pooling='
avg')
pretrained_model3.trainable = False


inputs3 = pretrained_model3.input
x3 = tf.keras.layers.Dense(128, activation='relu')(pretrained_
model3.output)
outputs3 = tf.keras.layers.Dense(3, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
model.compile(loss = 'categorical_crossentropy', optimizer='ad
am', metrics=['accuracy'])


model.summary()

# plot the model including the sizes of the model
tf.keras.utils.plot_model(model, show_shapes=True)


his=model.fit(datagen.flow(trainx,trainy,batch_size=32),valida
tion_data=(testx,testy),epochs=30)
```

3. Arsitektur MobileNet

```
pretrained_model3 = tf.keras.applications.InceptionV3(input_sh
ape=(299,299,3),include_top=False,weights='imagenet',pooling='
avg')
pretrained_model3.trainable = False


inputs3 = pretrained_model3.input
x3 = tf.keras.layers.Dense(128, activation='relu')(pretrained_
model3.output)
outputs3 = tf.keras.layers.Dense(3, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
model.compile(loss = 'categorical_crossentropy', optimizer='ad
am', metrics=['accuracy'])


model.summary()

# plot the model including the sizes of the model
tf.keras.utils.plot_model(model, show_shapes=True)


his=model.fit(datagen.flow(trainx,trainy,batch_size=32),valida
tion_data=(testx,testy),epochs=30)
```

D. Evaluasi Kinerja Model

```
y_pred=model.predict(testx)
pred=np.argmax(y_pred,axis=1)
ground = np.argmax(testy,axis=1)
print(classification_report(ground,pred))


get_acc = his.history['accuracy']
value_acc = his.history['val_accuracy']
get_loss = his.history['loss']
validation_loss = his.history['val_loss']

epochs = range(len(get_acc))
plt.plot(epochs, get_acc, 'r', label='Accuracy of Training dat
a')
plt.plot(epochs, value_acc, 'b', label='Accuracy of Validation
 data')
plt.title('Training vs validation accuracy')
plt.legend(loc=0)
plt.figure()
```

```
plt.show()


epochs = range(len(get_loss))
plt.plot(epochs, get_loss, 'r', label='Loss of Training data')
plt.plot(epochs, validation_loss, 'b', label='Loss of Validati
on data')
plt.title('Training vs validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()

fpr, tpr, thresholds = roc_curve(testy.ravel(), y_pred.ravel()
)
auc_ = auc(fpr, tpr)

plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='ROC (area = {:.3f})'.format(auc_))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()


from sklearn.metrics import confusion_matrix
import seaborn as sns
conf = confusion_matrix(ground, pred)
fig, ax = plt.subplots(figsize=(7,7))
sns.heatmap(conf, annot= True, ax = ax)

ax.set_xlabel('Predict labels')
ax.set_title('Confusion Matrix')


# Menampilkan kurva AUC-ROC
Y_pred = model.predict(test_x)

n_classes = 3
lw = 2
Y_val = test_y
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
```

```
    fpr[i], tpr[i], _ = roc_curve(Y_val[:, i], Y_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(Y_val.ravel(), Y_pre
d.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])


# Pertama-tama, gabungkan semua False Positive Rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_cl
asses)]))

# Kemudian interpolasi semua kurva ROC pada titik ini
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Terakhir, rata-rata dan hitung AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot semua kurva ROC
plt.figure(figsize=(8, 8))
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['orange', 'green', 'blue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic
 to multi-class')
plt.legend(loc="lower right")
plt.show()
```

E. Convert model for deployment

```
model = tf.keras.models.load_model('/content/drive/MyDrive/cts
can/modelxception.h5')


model.save('/content/drive/MyDrive/ctscan/modelxception.h5')


# Convert the model.
model = tf.keras.models.load_model('/content/drive/MyDrive/cts
can/modelxception.h5')
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
tflite_model_file = '/content/model.tflite'

with open(tflite_model_file, "wb") as f:
    f.write(tflite_model)
```