DAFTAR PUSTAKA

Astuti, L. W. (2019). *Ekstrasi Fitur Citra MRI Otak Menggunakan Data Wavelet Transform (DWT) untuk Klasifikasi Penyakit Tumor Otak*. Jurnal Ilmiah Informatika Global Volume 10 No. 02, 7.

Buckner, Cameron. (2018). *Empiricism without Magic: Transformational Abstraction in Deep Convolutional Neural Networks. Synthese*. 195. 10.1007/s11229-018-01949-1.

Cinar, A. & Yildirim, M. (2020) "*Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture*," Med. Hypotheses, vol. 139.

DocDoc. (2020, Mei 1). *DocDoc*. Diakses pada 8 Agustus 2022, dari DocDoc: *https://www.docdoc.com/id/info/body/brain*

Geraldy, C., & Lubis, C. (2020). *Pendeteksian Dan Pengenalan Jenis Mobil Menggunakan Algoritma You Only Look Once Dan Convolutional Neural Network*. Jurnal Ilmu Komputer dan Sistem Informasi, 3.

Han, Jun; Morag, Claudio (1995). "*The influence of the sigmoid function parameters on the speed of backpropagation learning*". In Mira, José; Sandoval, Francisco (eds.). From Natural to Artificial Neural Computation. Lecture Notes in Computer Science. Vol. 930. pp. 195–201. doi:10.1007/3-540-59497-3_175. ISBN 978-3-540-59497-0.

Hashemzehi, S., Javad, S., Mahdavi, M., & Kamel, S. R. *"Detection of brain tumors from MRI images base on deep learning using hybrid model CNN and NADE,"* Biocybern. Biomed. Eng., vol. 40, no. 3, pp. 1225–1232, 2020.

Hello Sehat. (2015, Juni 3). *hello sehat*. Diakses pada 8 Agustus 2022, dari hello sehat: *https://hellosehat.com/kanker/kanker-otak/tumor-otak/*

Heranurweni, S., Destyningtias, B., & Nugroho, A. K. (2018). *Klasifikasi Pola Image Pada Pasien Tumor Otak Berbasis Jaringan Syaraf Tiruan (Studi Kasus Penanganan Kuratif Pasien Tumor Otak).* eLEKTRIKAL, Vol. 10No.2Tahun2018; hal 37-40, 4.

Hidayatullah, R. (2021). *Klasifikasi Tumor Otak Menggunakan Convolutional Neural Network Dengan Arsitektur EfficientNet B0.* Pekanbaru: Redho Hidayatullah.

Jajang, S., & Riffa, H. (2019). *Klasifikasi Hasil Citra MRI Otak Untuk Memprediksi Jenis Tumor Otak Dengan Metode Image threshold dan GLCM Menggunakan Algoritma K-NN (Nearest Neighbor) Classifier Berbasis Web*. Jurnal Infotronik Volume 4, 6.

Kristanti, I. A. (2021). *Classification Of Brain Tumor On Brain MRI Images Using Convolutional Neural Network Method.* Tugas Akhir thesis, 6.

Medium. (2018, Februari 24). *Medium.* Diakses pada 8 Agustus 2022, dari: https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

Nurhikmat, T. (2018). *Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Wayang Golek.* Tugas Akhir.

Pamungkas, A. (2017). Pengolahan Citra Digital. Diakses pada 8 Agustus 2022, dari : pemrogramanmatlab.com/2017/07/26/pengolahan-citra-digital/

Qodri, K. N., Soesanti, I., & Nugroho, H. A. (2021). *Image Analysis for MRI-Based Brain Tumor Classification.* IJITEE, Vol. 5, No. 1, 8.

Radical, R. R., Fetty, T. A., & Budi, N. (2020). *Implementasi Metode Extreme Learning Machine untuk Klasifikasi Tumor Otak pada Citra Magnetic Resonance Imaging*. Seminar Nasional Informatika Bela Negara (SANTIKA), 5.

Rahmawati, A. (2020). *Klasifikasi Tumor Otak Menggunakan Convolutional Neural Network*.(hal. 34). Malang: Teknik Informatika Universitas Muhammadiyah Malang.

Soesanti, I., Susanto, A., Widodo, T. S., & Tjokronagoro, M. (2011). *Ekstraksi Ciri dan Identifikasi Citra Otak MRI Berbasis Eigenbrain Image*. ekstraksi Ciri dan Identifikasi Citra Otak MRI, 6.

Srinivasan, K. (2021). *Performance Comparison of Deep CNN Models for Detecting Driver's Distraction*. Computers, Materials & Continua, 68(3).

Gongting, T. (2019, Mar 29). *Medium*. Diakses pada 8 November 2022, dari Medium : *medium.com/@penggongting/understanding-roc-auc-pros-and-cons-why-is-bier-score-a-great-supplement-c7a0c976b679*

Wulandari, I., Yasin, H., & Widiharih, T. (2020). *Klasifikasi Citra Digital Bumbu Dan Rempah Dengan Algoritma Convolutional Neural Network (CNN).* Jurnal Gaussian, 9(3), 273-282.

LAMPIRAN

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

import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
import tensorflow as tf
from PIL import Image
from numpy import asarray
from keras import applications
from tensorflow.keras import layers
from tensorflow.keras import Model
from scipy import interp
from itertools import cycle
from tensorflow.keras.applications import NASNetLarge
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.model_selection import train_test_split

directory = '/content/drive/MyDrive/BrainTumor3'

File=[]
for file in os.listdir(directory):
   File+=[file]
print(File)
directoryShow      =      "/content/drive/MyDrive/BrainTumor3/pituitary_tumor/p
(133).jpg"
img = mpimg.imread(directoryShow)
imgplot = plt.imshow(img)
plt.show()
```

```
img_resize = load_img(directoryShow, grayscale=False, color_mode='rgb', target_size=(224,224))
plt.figure(figsize=(3,3))
imgplot = plt.imshow(img_resize)

image1 = img_to_array(img_resize)

image1

img_norm = image1/255.0

img_norm

dataset=[]
mapping={'no_tumor':0, 'pituitary_tumor':1, 'meningioma_tumor':2, 'glioma_tumor':3}
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=False, color_mode='rgb', target_size=(224,224))
        image=img_to_array(image)
        image=image/255.0
        dataset.append([image,count])
    count=count+1

len(dataset)

data,labels0=zip(*dataset)

labels1=to_categorical(labels0)
data=np.array(data)
labels=np.array(labels1)
print(data.shape)
print(labels.shape)

data.shape

trainx,testx,trainy,testy=train_test_split(data,labels,test_size=0.2,random_state=44)

len(trainx), len(testx)

print(trainx.shape)
```

```
print(testx.shape)
print(trainy.shape)
print(testy.shape)

MobileNet

datagen =
ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=20,z
oom_range=0.2,

width_shift_range=0.2,height_shift_range=0.2,shear_range=0.1,fill_mode="neare
st")

pretrained_model3 =
tf.keras.applications.MobileNet(input_shape=(224,224,3),include_top=False,weig
hts='imagenet')
pretrained_model3.trainable = False

inputs3 = pretrained_model3.input
x3 = tf.keras.layers.GlobalAveragePooling2D()(pretrained_model3.output)
x3 = tf.keras.layers.Dense(512, activation='relu')(x3)
x3 = tf.keras.layers.Dense(1024, activation='relu')(x3)
x3 = tf.keras.layers.Dropout(0.8)(x3)
outputs3 = tf.keras.layers.Dense(4, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
model.compile(loss = 'categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.summary()

tf.keras.utils.plot_model(model, show_shapes=True)

his=model.fit(datagen.flow(trainx,trainy,batch_size=32),validation_data=(testx,tes
ty),epochs=30)

model.evaluate(testx,testy)
model.save('/content/drive/MyDrive/modelmobilenet.h5')

model  = tf.keras.models.load_model('/content/drive/MyDrive/modelmobilenet.h5')

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 data')
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 Validation data')
plt.title('Training vs validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()

y_true = np.argmax(testy, axis=1)
y_true

conf_brain_tumor = confusion_matrix(y_true=y_true, y_pred=y_pred)

Y_pred = model.predict(testx)

n_classes = 4
lw = 2
Y_val = testy
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_pred.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_classes)]))

# 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()

load_img("/content/drive/MyDrive/BrainTumor3/meningioma_tumor/m
(137).jpg",target_size=(224,224))

image=load_img("/content/drive/MyDrive/BrainTumor3/meningioma_tumor/m
(137).jpg",target_size=(224,224))

image=img_to_array(image)
```

```
image=image/255.0
prediction_image=np.array(image)
prediction_image= np.expand_dims(image, axis=0)

reverse_mapping={0:'glioma_tumor', 1:'meningioma_tumor', 2:'no_tumor',
3:'pituitary_tumor'}

def mapper(value):
    return reverse_mapping[value]

prediction=model.predict(prediction_image)
value=np.argmax(prediction)
move_name=mapper(value)
print("Prediction is {}.".format(move_name))
```

DENSENET201

```
pretrained_model3 =
tf.keras.applications.DenseNet201(input_shape=(224,224,3),include_top=False,w
eights='imagenet')
pretrained_model3.trainable = False

inputs3 = pretrained_model3.input
x3 = tf.keras.layers.GlobalAveragePooling2D()(pretrained_model3.output)
x3 = tf.keras.layers.Dense(512, activation='relu')(x3)
x3 = tf.keras.layers.Dense(1024, activation='relu')(x3)
x3 = tf.keras.layers.Dropout(0.8)(x3)
outputs3 = tf.keras.layers.Dense(4, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
model.compile(loss = 'categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.summary()

his=model.fit(datagen.flow(trainx,trainy,batch_size=32),validation_data=(testx,tes
ty),epochs=30)

model.evaluate(testx,testy)

model.save('/content/drive/MyDrive/modeldensenet201.h5')

model_densenet201 =
tf.keras.models.load_model('/content/drive/MyDrive/modeldensenet201.h5')

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 data')
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 Validation data')
plt.title('Training vs validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()

y_pred = model.predict(np.expand_dims(testx,-1))
y_pred = np.argmax(y_pred, axis=1)
# y_test = np.expand_dims(y_test,-1)
y_true = np.argmax(testy, axis=1)
y_true

conf_brain_tumor = confusion_matrix(y_true=y_true, y_pred=y_pred)

fig, ax = plt.subplots(figsize=(7,7))
sns.heatmap(conf_brain_tumor, annot= True, ax = ax)

ax.set_xlabel('Predict labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['glioma_tumor','meningioma_tumor',          'no_tumor',
'pituitary_tumor'])
ax.yaxis.set_ticklabels(['glioma_tumor','meningioma_tumor',          'no_tumor',
'pituitary_tumor'])

Y_pred = model_densenet201.predict(testx)

n_classes = 4
```

```
lw = 2
Y_val = testy
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_pred.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_classes)]))

# 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()
```

NASNETLARGE

```
dataset=[]
mapping={'no_tumor':0, 'pituitary_tumor':1, 'meningioma_tumor':2, 'glioma_tumor':3}
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=False, color_mode='rgb', target_size=(331,331))
        image=img_to_array(image)
        image=image/255.0
        dataset.append([image,count])
    count=count+1

len(dataset)

data,labels0=zip(*dataset)

labels1=to_categorical(labels0)
data=np.array(data)
labels=np.array(labels1)
print(data.shape)
print(labels.shape)

trainx,testx,trainy,testy=train_test_split(data,labels,test_size=0.2,random_state=44)

print(trainx.shape)
print(testx.shape)
print(trainy.shape)
print(testy.shape)
```

```
datagen = ImageDataGenerator(horizontal_flip=True,vertical_flip=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")

pretrained_model3 = tf.keras.applications.NASNetLarge(input_shape=(331,331,3),include_top=False, weights='imagenet')
pretrained_model3.trainable = False

inputs3 = pretrained_model3.input
x3 = tf.keras.layers.GlobalAveragePooling2D()(pretrained_model3.output)
x3 = tf.keras.layers.Dense(512, activation='relu')(x3)
x3 = tf.keras.layers.Dense(1024, activation='relu')(x3)
x3 = tf.keras.layers.Dropout(0.8)(x3)
outputs3 = tf.keras.layers.Dense(4, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

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

model.evaluate(testx,testy)

model.save('/content/drive/MyDrive/modelNASNetLarge.h5')

model_NASNetLarge = tf.keras.models.load_model('/content/drive/MyDrive/modelNASNetLarge.h5')

tf.keras.utils.plot_model(model_inceptionresnetv2, show_shapes=True)

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 data')
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 Validation data')
plt.title('Training vs validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()

y_pred = model.predict(np.expand_dims(testx,-1))
y_pred = np.argmax(y_pred, axis=1)
# y_test = np.expand_dims(y_test,-1)
y_true = np.argmax(testy, axis=1)
y_true

conf_brain_tumor = confusion_matrix(y_true=y_true, y_pred=y_pred)

fig, ax = plt.subplots(figsize=(7,7))
sns.heatmap(conf_brain_tumor, annot= True, ax = ax)

ax.set_xlabel('Predict labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['glioma_tumor','meningioma_tumor',          'no_tumor',
'pituitary_tumor'])
ax.yaxis.set_ticklabels(['glioma_tumor','meningioma_tumor',          'no_tumor',
'pituitary_tumor'])

y_pred = model_NASNetLarge.predict(testx)

n_classes = 4
lw = 2
Y_val = testy
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_pred.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_classes)]))

# 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()
```