LAPORAN PENGGANTI UAS

MK KECERDASAN BUATAN

DIKUMPULKAN PADA: 28 DESEMBER 2022

Nama Mahasiswa	**: Muhammad Aidan Daffa Junaidi**
NPM	**: 1906300800**
Judul Paper	**: Optic-Net: A Novel Convolutional Neural Network for Diagnosis of Retinal Diseases from Optical Tomography Images**
Penulis	**: Sharif Amit Kamran, Sourajit Saha, Ali Shihab Sabbir, dan Alireza Tavakkoli**
Dipublikasikan di	**: Cornell University Library, arXiv.org**
Tahun	**: 2019**
Modality/Alat Pencitraan	**: Optical Coherence Tomography (OCT)**
Task/Tujuan pengolahan	**: Mendiagnosis berbagai penyakit retina dari gambar Spectral Domain Optical Coherence Tomography (SD-OCT).**
Bahasa Pemrograman	**: Python**
Link Paper	**: https://arxiv.org/pdf/1910.05672v1.pdf**
Link Source Code	**: https://github.com/SharifAmit/OpticNet-71**

RESUME PAPER

1 PENDAHULUAN

Jelaskan dalam Bahasa Indonesia:

a) latar belakang masalah/topik penelitian yang diangkat,

Mendesain aristektur algoritma CNN yang mampu mendiagnosis beberpa penyakit retina dari gambar Spectral Domain Optical Coherence Tomography (SD-OCT) secara optimal.

b) urgensi topik penelitian,

Saat ini berbagai pendekatan dengan image processing, machine learning, dan algoritma deep learning telah digunakan untuk mendeteksi dan mendiagnosis penyakit retina. Sayangnya, pendekatan – pendekatan ini masih rentan terhadap kesalahan (error) dan ketidakefisienan komputasi, yang hasilnya masih memerlukan intervensi lebih lanjut dari pakar manusia. Didalam paper yang digunakan umtuk tugas ini, diusulkan arsitektur CNN baru yang berhasil membedakan antara berbagai degenerasi lapisan retina dan penyebab dasarnya. Yang hasilnnya akan memperbagus akurasi.

c) metode-metode terkini (state-of-the-art) yang telah dihasilkan oleh peneliti lain yang disebutkan dalam paper tsb,

Nomor referensi	Penulis	Judul Paper	Metode yang diusulkan	Kelebihan yang diusung	Celah/Kekurangan
6	K. Alsaih, G. Lemaitre, M. Rastgoo, J. Massich, D. Sidibe, dan F. Meriaudeau	Machine learning techniques for diabetic macular edema (dme) classification	Menggunakan generic pipelinetermasuk preprocessing, deteksi fitur, representasi fitur, dan klasifikasi..	Selain membandingkan fitur individu dan gabungan, pendekatan representasi yang berbeda dan pengklas	Dataset relatif sedikit, accuracy masih bisa ditingkatkan, belum menggunakan arsitektur deep learning

		on sd-oct images		ifikasi yang berbeda juga dievaluasi	
26	M. Treder, J. L. Lauermann, and N. Eter	"Automated detection of exudative age-related macular degeneration in spectral domain optical coherence tomography using deep learning	Terdapat healthy control group dalam menyiapkan data dan menggunakan Deep Learning CNN	Hasil dari evaluasi metric sangat bagus	Hanya mendeteksi penyakit AMD
34	M. Awais, H. Muller, T. B. Tang, and F. Meriaudeau	Classification of sd-oct images using a deep learning approach	Menggunakan algoritma CNN, arsitektur VGG 16	Penerapan VGG berhail meningkatkan metric evaluasi SE sebesar 20%	Dataset relatif sedikit , Metric evaluasi masih terbilang kecil
35	S. P. K. Karri, D. Chakraborty, and J. Chatterjee	Transfer learning based classification of optical coherence tomography images with diabetic macula	Menggunakan algoritma CNN, arsitektur GoogLeNet dan menggunakan transfer learning dari imagenet	penggunaan model pre-trained untuk konvergensi yang lebih cepat dengan lebih sedikit data	Dataset relatif sedikit , Metric evaluasi masih bisa ditingkatkan

		r edema and dry age-related macular degeneration			
	https://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf				

d) ide utama dari metode yang penulis usulkan,

Penulis mengusulkan arsitektur baru yang mengungguli model klasifikasi lain sambil menangani masalah *gradient explosion*. Pendekatan yang penulis lakukan mencapai akurasi hampir sempurna 99,8% dan 100% untuk dua set data Retinal SD-OCT yang tersedia secara terpisah. penulis mengusulkan jaringan CNN yang berspesialisasi dalam mengidentifikasi penyakit retina dengan presisi yang hampir sempurna.

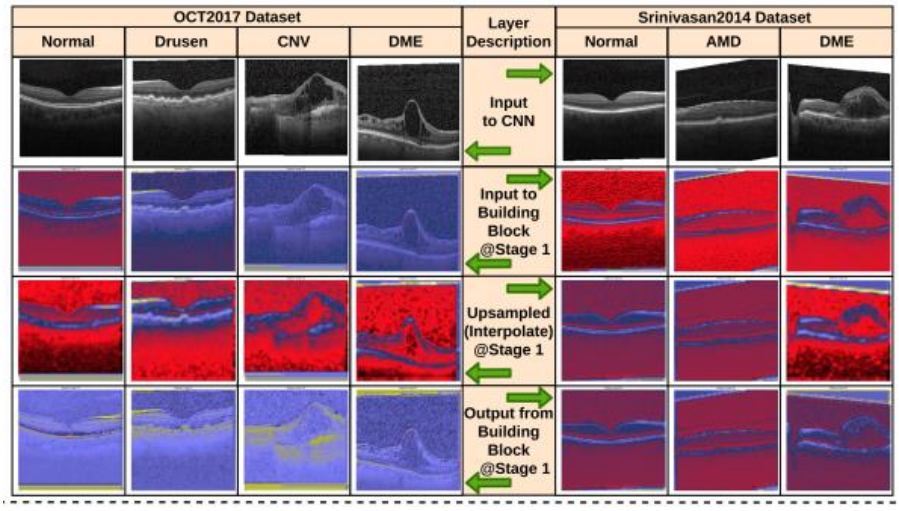
e) tujuan penelitian/kontribusi/kebaruan yang ditawarkan.

Melalui arsitektur ini penulis mengusulkan (a) unit residu baru yang memasukkan Atrous Separable Convolution, (b) building block baru dan (c) mekanisme untuk mencegah degradasi gradien.

2 METODE YANG DITAWARKAN

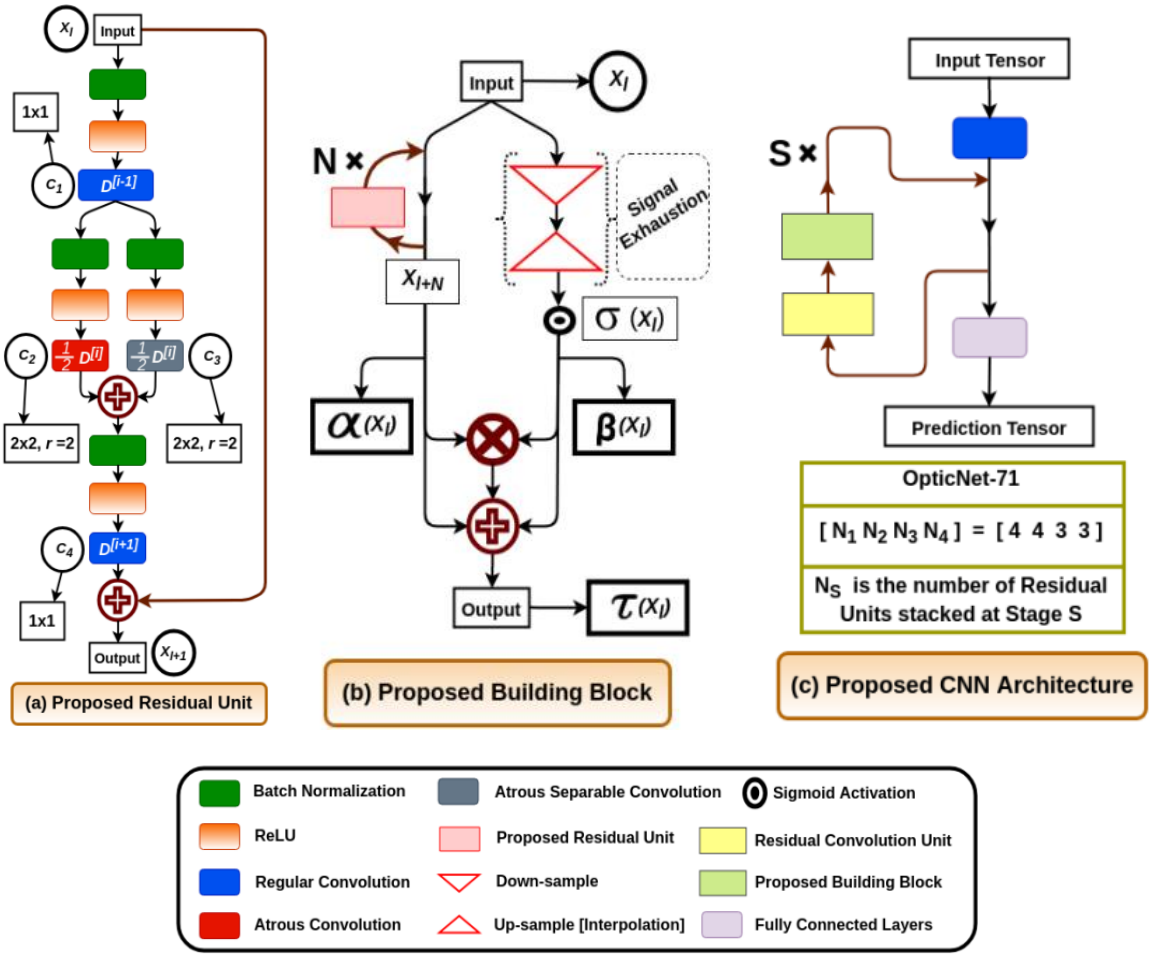
2.1) Prinsip dasar pengolahan sinyal/citra (yang dipelajari pada kuliah ini) yang diterapkan pada metode tsb

Prinsip yang digunakan pada penelitian ini adalah menggunakan algoritma CNN. Jadi gambar akan mengalami proses Konvolusi, Pooling, Interpolation dan lainnya. Semua proses tersebut terjadi didalam building block dan arsitektur bagian residual unit yang akan ditambilkan pada section Algoritma



2.2) Algoritma dan/atau alur kerja metode tsb

Flowchart dibawah ini merupakan arsitektur yang menjadi keterbaharuan paper ini, yaitu membangun arsitektur OpticNet. Arsitektur ini diharapkan dapat menyelesaikan permasalahan gradient explosion dan gradient degradation.



EKSPERIMEN DAN ANALISIS

1 DATA

[Jelaskan mengenai data yang digunakan, sumber data, jenis citra, dll]

Sumber dataset :

- OCT 2017
  Link : https://data.mendeley.com/datasets/rscbjbr9sj/3
  Source : University of California San Diego, Guangzhou Women and Children's Medical Center
- Srinivasan2014
  Link : https://people.duke.edu/~sf59/Srinivasan_BOE_2014_dataset.htm
  Source : DUKE UNIVERSITY

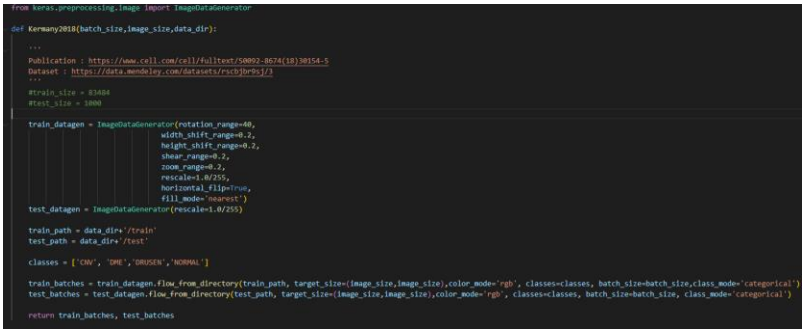
Pengambilan gambar menggunakan alat citra beruuupa Optical Coherence Tomography (OCT) Scan Penelitian ini menggunakan 2 dataset yang dijalankan secara terpisah. Nama dari dataset yang pertama adalah OCT2017 dan yang kedua adalah Srinivasan2014. Dataset OCT2017 terdapat Kumpulan data ini berisi ribuan gambar OCT dan X-Ray Dada tervalidasi yang dideskripsikan dan dianalisis dalam "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning". Gambar dibagi menjadi satu set pelatihan dan satu set pengujian pasien independen. Gambar diberi label sebagai (penyakit)-(ID pasien acak)-(nomor gambar oleh pasien ini) dan dibagi menjadi 4 direktori: CNV, DME, DRUSEN, dan NORMAL. Sedangkan pada dataset Srinivasan2014 terdiri dari pemindaian volumetrik yang diperoleh dari 45 pasien: 15 pasien normal, 15 pasien dengan AMD kering, dan 15 pasien dengan DME menggunakan Spectralis SD-OCT

Jadi pada dataset OCT2017 terdapat gambar dengan 4 label, yaitu CNV, DME, DRUSEN, dan NORMAL dan pada dataset Srinivasan2014 terdapat 3 label, yaitu normal, DME, dan AMD.

2 KODE

[Jelaskan function-function utama yang dipakai, jelaskan perubahan yang ditambahkan pada kode (jika ada). Lampirkan kode pada file terpisah lengkap dengan data.]

DataLoader.py

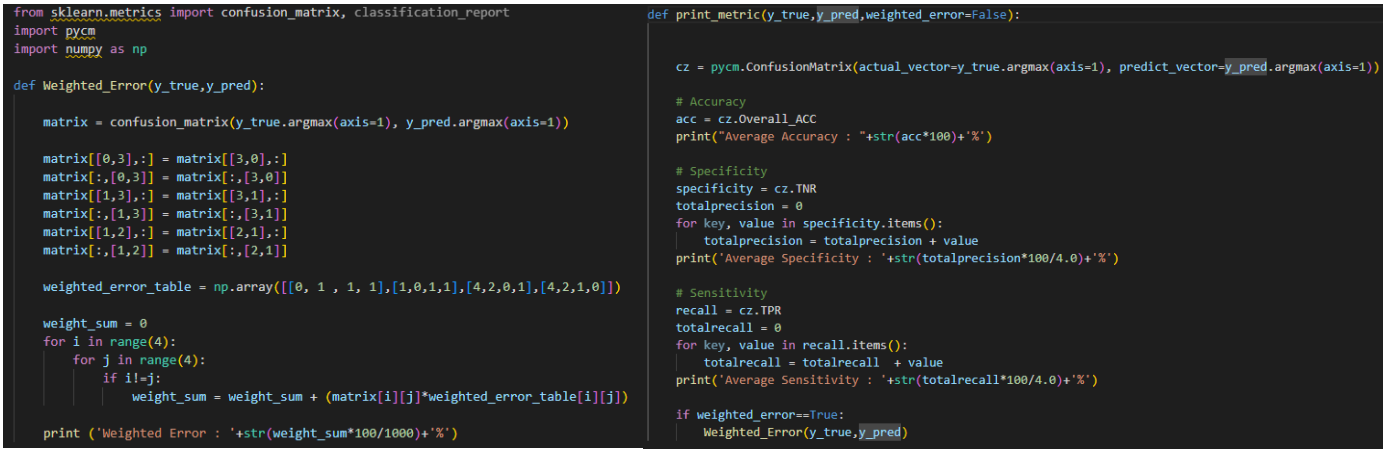


```
from keras.preprocessing.image import ImageDataGenerator

def Kermany2018(batch_size,image_size,data_dir):
    '''
    Publication : https://www.cell.com/cell/fulltext/S0092-8674(18)30154-5
    Dataset : https://data.mendeley.com/datasets/rscbjbr9sj/3
    '''
    #train_size = 83484
    #test_size = 1000

    train_datagen = ImageDataGenerator(rotation_range=40,
                                       width_shift_range=0.2,
                                       height_shift_range=0.2,
                                       shear_range=0.2,
                                       zoom_range=0.2,
                                       rescale=1.0/255,
                                       horizontal_flip=True,
                                       fill_mode='nearest')
    test_datagen = ImageDataGenerator(rescale=1.0/255)

    train_path = data_dir+'/train'
    test_path = data_dir+'/test'

    classes = ['CNV', 'DME','DRUSEN','NORMAL']

    train_batches = train_datagen.flow_from_directory(train_path, target_size=(image_size,image_size),color_mode='rgb', classes=classes, batch_size=batch_size,class_mode='categorical')
    test_batches = test_datagen.flow_from_directory(test_path, target_size=(image_size,image_size),color_mode='rgb', classes=classes, batch_size=batch_size, class_mode='categorical')

    return train_batches, test_batches
```

```
def Srinivasan2014(batch_size,image_size,data_dir):
    '''
    Publication : http://www.opticsinfobase.org/boe/abstract.cfm?uri=boe-5-10-3568
    Dataset : http://people.duke.edu/~sf59/Srinivasan_BOE_2014_dataset.htm
    '''
    #train_size = 2916
    #test_size = 315

    train_path = data_dir+'/Train'
    test_path = data_dir+'/Test'

    classes=['AMD', 'DME', 'NORMAL']

    train_datagen = ImageDataGenerator(rotation_range=40,
                                       width_shift_range=0.2,
                                       height_shift_range=0.2,
                                       shear_range=0.2,
                                       zoom_range=0.2,
                                       rescale=1.0/255,
                                       horizontal_flip=True,
                                       fill_mode='nearest')
    test_datagen = ImageDataGenerator(rescale=1.0/255)

    train_batches = train_datagen.flow_from_directory(train_path, target_size=(image_size,image_size),color_mode='rgb', classes=classes, batch_size=batch_size,class_mode='categorical')
    test_batches = test_datagen.flow_from_directory(test_path, target_size=(image_size,image_size),color_mode='rgb', classes=classes, batch_size=batch_size, class_mode='categorical')

    return train_batches, test_batches
```

Pada file ini terdapat 2 funtion yange namanya Kermany2018 dan Srinivasan2014. Kedua function memiliki tujuan yang sama, yaitu untuk me-load data, mengaugmentasi data yang tujuannya memperbanyak dataset, dan membagi data test dan train. Beda dari kedua function tersebut adalah function Kermany2018 untuk mengolah data OCT2017 sedangkan function Srinivasan2014 untuk mengolah data Srinivasan2014.

Metrics.py



```
from sklearn.metrics import confusion_matrix, classification_report
import pycm
import numpy as np

def Weighted_Error(y_true,y_pred):

    matrix = confusion_matrix(y_true.argmax(axis=1), y_pred.argmax(axis=1))

    matrix[[0,3],:] = matrix[[3,0],:]
    matrix[:,[0,3]] = matrix[:,[3,0]]
    matrix[[1,3],:] = matrix[[3,1],:]
    matrix[:,[1,3]] = matrix[:,[3,1]]
    matrix[[1,2],:] = matrix[[2,1],:]
    matrix[:,[1,2]] = matrix[:,[2,1]]

    weighted_error_table = np.array([[0, 1 , 1, 1],[1,0,1,1],[4,2,0,1],[4,2,1,0]])

    weight_sum = 0
    for i in range(4):
        for j in range(4):
            if i!=j:
                weight_sum = weight_sum + (matrix[i][j]*weighted_error_table[i][j])

    print ('Weighted Error : '+str(weight_sum*100/1000)+'%')
```

```
def print_metric(y_true,y_pred,weighted_error=False):


    cz = pycm.ConfusionMatrix(actual_vector=y_true.argmax(axis=1), predict_vector=y_pred.argmax(axis=1))

    # Accuracy
    acc = cz.Overall_ACC
    print("Average Accuracy : "+str(acc*100)+'%')

    # Specificity
    specificity = cz.TNR
    totalprecision = 0
    for key, value in specificity.items():
        totalprecision = totalprecision + value
    print('Average Specificity : '+str(totalprecision*100/4.0)+'%')

    # Sensitivity
    recall = cz.TPR
    totalrecall = 0
    for key, value in recall.items():
        totalrecall = totalrecall  + value
    print('Average Sensitivity : '+str(totalrecall*100/4.0)+'%')

    if weighted_error==True:
        Weighted_Error(y_true,y_pred)
```

Pada file ini terdapat 2 function. Kedua function ini bertujuan untuk mengolah metric evaluasi. Pada function pertama mengolah metric evaluasi WeightedError. Pada function kedua functionya bertujuan untuk mengolah evaluasi metric Accuracy, S Specificity, dan Sensitivity.

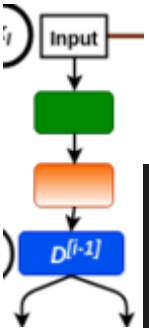
Model.py

```
import tensorflow as tf
import keras.backend
from keras.models import Model
from keras.layers import Input, Multiply,GlobalAveragePooling2D, Add, Dense, Activation, Maximum, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, Max
from keras.optimizers import Adam
from keras.initializers import glorot_uniform

def res_conv(X, filters, base, s):

    name_base = base + '/branch'

    F1, F2, F3 = filters

    ##### Branch1 is the main path and Branch2 is the shortcut path #####

    X_shortcut = X

    ##### Branch1 #####
    # First component of Branch1
    X = BatchNormalization(axis=-1, name=name_base + '1/bn_1')(X)
    X= Activation('relu', name=name_base + '1/relu_1')(X)
    X = Conv2D(filters=F1, kernel_size=(1,1), strides=(1,1), padding='valid', name=name_base + '1/conv_1', kernel_initializer=glorot_uniform(seed=0))(X)

    # Second component of Branch1
    X = BatchNormalization(axis=-1, name=name_base + '1/bn_2')(X)
    X = Activation('relu', name=name_base + '1/relu_2')(X)
    X = Conv2D(filters=F2, kernel_size=(2,2), strides=(s,s), padding='same', name=name_base + '1/conv_2', kernel_initializer=glorot_uniform(seed=0))(X)

    # Third component of Branch1
    X = BatchNormalization(axis=-1, name=name_base + '1/bn_3')(X)
    X = Activation('relu', name=name_base + '1/relu_3')(X)
    X = Conv2D(filters=F3, kernel_size=(1,1), strides=(1,1), padding='valid', name=name_base + '1/conv_3', kernel_initializer=glorot_uniform(seed=0))(X)

    ##### Branch2 ####
    X_shortcut = BatchNormalization(axis=-1, name=name_base + '2/bn_1')(X_shortcut)
    X_shortcut= Activation('relu', name=name_base + '2/relu_1')(X_shortcut)
    X_shortcut = Conv2D(filters=F3, kernel_size=(1,1), strides=(s,s), padding='valid', name=name_base + '2/conv_1', kernel_initializer=glorot_uniform(seed=0))(X_shortcut)

    # Final step: Add Branch1 and Branch2
    X = Add(name=base + '/Add')([X, X_shortcut])

    return X
```

```
def res_identity(X, filters, base):

    name_base = base + '/branch'

    F1, F2, F3 = filters

    ##### Branch1 is the main path and Branch2 is the shortcut path #####

    X_shortcut = X

    ##### Branch1 #####
    # First component of Branch1
    X = BatchNormalization(axis=-1, name=name_base + '1/bn_1')(X)
    Shortcut= Activation('relu', name=name_base + '1/relu_1')(X)
    X = Conv2D(filters=F1, kernel_size=(1,1), strides=(1,1), padding='valid', name=name_base + '1/conv_1', kernel_initializer=glorot_uniform(seed=0))(Shortcut)

    # Second component BranchOut 1
    X1 = BatchNormalization(axis=-1, name=name_base + '1/ConvBn_2')(X)
    X1 = Activation('relu', name=name_base + '1/ConvRelu_2')(X1)
    X1 = Conv2D(filters=F2, kernel_size=(2,2), dilation_rate=(2, 2),strides=(1,1), padding='same', name=name_base + '1/Conv_2', kernel_initializer=glorot_uniform(seed=0))(X1)

    # Second component BrancOut 2
    X2 = BatchNormalization(axis=-1, name=name_base + '1/SepBn_2')(X)
    X2 = Activation('relu', name=name_base + '1/SepRelu_2')(X2)
    X2 = SeparableConv2D(filters=F2, kernel_size=(2,2), dilation_rate=(2, 2),strides=(1,1), padding='same', name=name_base + '1/SepConv_2', kernel_initializer=glorot_uniform(seed=0))(X2)

    # Second component Add-BranchOut
    X = Add(name=base + '/Add-2branches')([X1, X2])

    # Third component of Branch1
    X = BatchNormalization(axis=-1, name=name_base + '1/bn_3')(X)
    X = Activation('relu', name=name_base + '1/relu_3')(X)
    X = Conv2D(filters=F3, kernel_size=(1,1), strides=(1,1), padding='valid', name=name_base + '1/conv_3', kernel_initializer=glorot_uniform(seed=0))(X)

    # Final step: Add Branch1 and the original Input itself
    X = Add(name=base + '/Add')([X_shortcut,X])

    return X
```

```
def EncoderDecoder(X, name_base):
    X = MaxPooling2D((3,3), strides=(2,2), padding='same', name = name_base + '/Downsample1')(X)
    #X = Conv2D(outgoing_depth, (2,2), strides=(1,1), dilation_rate=(2,2), padding='same', name = name_base + '/DC1', kernel_initializer=glorot_uniform(seed=0))(X)
    X = UpSampling2D(size=(2, 2),interpolation='bilinear',name = name_base + '/Upsample1')(X)
    X = Activation('sigmoid', name = name_base + '/Activate')(X)
    return X
def RDBI(X, filters, base, number):

    for i in range(number):
        X = res_identity(X, filters, base+ '/id_'+str(1+i))

    return X
```

Pada file model.py merupakan file yang bertujuan dalam pembangunan arsitektur CNN yang akan digunakan dalam klasifikasi penyakit retina yang diusung oleh peneliti, yaitu OpticNet. Contoh jika kita mencuplik pada gambar flowchart arsitektur dan sebagian kode yang ada di file model.py yang ditunjukan dibawah ini




```
##### Branch1 #####
# First component of Branch1
X = BatchNormalization(axis=-1, name=name_base + '1/bn_1')(X)
X= Activation('relu', name=name_base + '1/relu_1')(X)
X = Conv2D(filters=F1, kernel_size=(1,1), strides=(1,1), padding='va
```

Pada gambar diatas jelas bahwa input akan melewati batch normalization, kemudian activation layer, kemudian conv layer.

```
def OpticNet(input_size,num_of_classes):
    input_shape=(input_size, input_size, 3) # Height x Width x Channel
    X_input = Input(input_shape)

    X = Conv2D(64, (7,7), strides=(2,2), padding='same', name ='CONV1', kernel_initializer=glorot_uniform(seed=0))(X_input)
    X = BatchNormalization(axis=-1, name ='BN1')(X)
    X = Activation('relu', name ='RELU1')(X)

    X = res_conv(X, [64,64,256], 'RC0', 1)

    # MID 1

    X1 = EncoderDecoder(X, 'EncoderDecoder1')

    X2 = RDBI(X, [32,32,256], 'RDBI1',4)

    X = Multiply(name = 'Mutiply1')([X1,X2])

    X = Add(name = 'Add1')([X,X1,X2])

    X = res_conv(X, [128,128,512], 'RC1', 2)

    # MID 2

    X1 = EncoderDecoder(X, 'EncoderDecoder2')

    X2 = RDBI(X, [64,64,512], 'RDBI2',4)

    X = Multiply(name = 'Mutiply2')([X1,X2])

    X = Add(name = 'Add2')([X,X1,X2])

    X = res_conv(X, [256,256,1024], 'RC2', 2)

    # MID 3

    X1 = EncoderDecoder(X, 'EncoderDecoder3')

    X2 = RDBI(X, [128,128,1024], 'RDBI3',3)

    X = Multiply(name = 'Mutiply3')([X1,X2])

    X = Add(name = 'Add3')([X,X1,X2])

    X = res_conv(X, [512,512,2048], 'RC3', 2)
```

```
# MID 4

X1 = EncoderDecoder(X, 'EncoderDecoder4')

X2 = RDBI(X, [256,256,2048], 'RDBI4',3)

X = Multiply(name = 'Mutiply4')([X1,X2])

X = Add(name = 'Add4')([X,X1,X2])


X = GlobalAveragePooling2D(name='global_avg_pool')(X)
X = Dense(256, name='Dense_1')(X)
X = Dense(num_of_classes, name='Dense_2')(X)
X = Activation('softmax', name='classifier')(X)


model = Model(inputs=X_input, outputs=X, name='')

model.compile(Adam(lr=.0001), loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()

return model
```

Utils.py

```python
from keras import callbacks

def callback_for_training(tf_log_dir_name='./log/',patience_lr=10,snapshot_name=None):
    cb = [None]
    """
    Tensorboard log callback
    """
    tb = callbacks.TensorBoard(log_dir=tf_log_dir_name, histogram_freq=0)
    cb[0]= tb


    """
    Early Stopping callback
    """
    #Uncomment for usage
    # early_stop = callbacks.EarlyStopping(monitor='val_acc', min_delta=0, patience=5, verbose=1, mode='auto',save_best_only=True)
    # cb.apppend(early_stop)

    """
    Model Checkpointer
    """
    if snapshot_name != None:
        checkpointer = callbacks.ModelCheckpoint(filepath="optic-net.{epoch:02d}-{val_acc:.2f}.hdf5",
                                verbose=0,
                                monitor='val_acc')
    else :
        checkpointer = callbacks.ModelCheckpoint(filepath=snapshot_name+".{epoch:02d}-{val_acc:.2f}.hdf5",
                                verbose=0,
                                monitor='val_acc')
    cb[1] = checkpointer

    """
    Reduce Learning Rate
    """
    reduce_lr_loss = callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=6, verbose=1, min_lr=1e-8, mode='auto')
    cb[2] = reduce_lr_loss

    return cb
```

Pada file ini terdapat 1 function yaitu callback_for_training. Function ini berfungsi untuk Menulis log TensorBoard setelah setiap kumpulan untuk memonitor metrik. Yang nantinya bisa di set untuk early stopping.

Visualize.py

```python
import matplotlib.pyplot as plt


def  plot_loss_acc(history,snapshot_name=None):
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(len(loss))
    plt.plot(epochs, loss, 'bo')
    plt.plot(epochs, val_loss, 'g')
    plt.title('Training and validation loss')
    plt.legend(['train', 'val'], loc='upper right')
    if snapshot_name == None:
        filename= 'OpticNet_loss.png'
    else:
        filename= snapshot_name+'_loss.png'
    plt.savefig(filename)
    plt.show()

    acc = history.history['acc']
    val_acc = history.history['val_acc']
    epochs = range(len(acc))
    plt.plot(epochs, acc, 'b')
    plt.plot(epochs, val_acc, 'g')
    plt.title('Training and validation accuracy')
    plt.legend(['train', 'val'], loc='lower right')
    if snapshot_name == None:
        filename= 'OpticNet_acc.png'
    else:
        filename= snapshot_name+'_acc.png'
    plt.savefig(filename)
    plt.show()
```

Pada file visualize.py terdapat function yang bertujuan untuk membuat grafik loss dan accuracy.

Train.py

```python
import argparse
import keras
from src.dataloader import Kermany2018, Srinivasan2014
from src.model import OpticNet
import time
import keras.backend as K
import gc
from src.utils import callback_for_training
from src.visualize import plot_loss_acc
from keras.models import load_model

def train(data_dir, logdir, input_size, dataset, batch_size, weights, epoch, pre_trained_model,snapshot_name):

    if dataset=='Srinivasan2014':
        train_batches, test_batches = Srinivasan2014(batch_size, input_size, data_dir)
        num_of_classes = 3
        train_size = 2916
        test_size = 315
    elif dataset == 'Kermany2018' :
        train_batches, test_batches = Kermany2018(batch_size, input_size, data_dir)
        num_of_classes = 4
        train_size = 83484
        test_size = 1000
    # Clear any outstanding net or memory
    K.clear_session()
    gc.collect()

    # Calculate the starting time

    start_time = time.time()

    # Callbacks for model saving, adaptive learning rate
    cb = callback_for_training(tf_log_dir_name=logdir,snapshot_name=snapshot_name)


    # Loading the model
    if weights == None:
        model = OpticNet(input_size,num_of_classes)
    else :
        model = load_model(weights)

    # Training the model
    history = model.fit_generator(train_batches, shuffle=True, steps_per_epoch=train_size //batch_size, validation_data=test_batches, validation_steps= test_size//batch_size, epochs=epoch, verbose=1, callbacks=cb)


    end_time = time.time()

    print("--- Time taken to train : %s hours ---" % ((end_time - start_time)//3600))
```

```python
    # Saving the final model
    if snapshot_name == None :
        model.save('OpticNet.h5')

    else :
        model.save(snapshot_name+'.h5')

    plot_loss_acc(history,snapshot_name)

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--dataset', type=str, required=True, help='Choosing between 2 OCT datasets', choices=['Srinivasan2014','Kermany2018'])
    parser.add_argument('--batch', type=int, default=8)
    parser.add_argument('--input_dim', type=int, default=224)
    parser.add_argument('--datadir', type=str, required=True, help='path/to/data_directory')
    parser.add_argument('--epoch', type=int, default=30)
    parser.add_argument('--logdir', type=str)
    parser.add_argument('--weights', type=str,default=None, help='Resuming training from previous weights')
    parser.add_argument('--model',type=str, default=None,help='Pretrained weights for transfer learning',choices=['ResNet50',
                                                'MobileNetV2','Xception'])
    parser.add_argument('--snapshot_name',type=str, default=None, help='Name the saved snapshot')
    args = parser.parse_args()
    train(args.datadir, args.logdir, args.input_dim, args.dataset, args.batch, args.weights, args.epoch, args.model, args.snapshot_name)
```

File ini bertujuan untuk mentraining data. Jika dataset yang sedang digunakan adalah OCT2017 maka akan memanggil function Kermany2018 yang berada di file dataloader.py . kemudian dilanjut dengan membersihkan session demi tujuan menyiapkan memory. Kemudian terdapat kode untuk memulai waktu dan memanggil function callback_for_training yang ada di file utils.py yang berguna untuk menyimpan model dan adaptive learning rate. Kemudian dilanjut dengan menload model dengan memanggil function yang ada di model.py. kemudian untuk mengolah metric evaluasi menggunakan kode pada file. Metrics.py. dan jika ingin membuat grafik menggunakan file visualize.py.

Test.py

```python
import tensorflow as tf
import keras
import argparse
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
import keras.backend as K
from src.metrics import print_metric
import time

def test(datadir,dataset,weights):

    test_datagen = ImageDataGenerator(rescale=1.0/255)
    image_size = 224

    if dataset=='Srinivasan2014':
        classes=['AMD', 'DME','NORMAL']
        batch = 315
        test_batches = test_datagen.flow_from_directory(datadir, target_size=(image_size,image_size),color_mode='rgb', classes=classes, b
    else:
        classes = ['CNV', 'DME','DRUSEN','NORMAL']
        batch=1000
        test_batches = test_datagen.flow_from_directory(datadir, target_size=(image_size,image_size),color_mode='rgb', classes=classes, b

    imgs, y_true = next(test_batches)

    K.clear_session()


    model = load_model(weights)

    start= time.time()
    y_pred = model.predict(imgs)
    end = time.time()

    print ((end-start)/1000)
    if dataset=='Srinivasan2014':
        print_metric(y_true,y_pred,weighted_error=False)
    else:
        print_metric(y_true,y_pred,weighted_error=True)
```

```python
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--dataset', type=str, required=True, help='Choosing between 2 OCT datasets', choices=['Srinivasan2014','Kermany2018'])
    parser.add_argument('--datadir', type=str, required=True, help='path/to/data_directory')
    parser.add_argument('--weights', type=str, required=True, help='Resuming training from previous weights')
    args = parser.parse_args()
    test(args.datadir, args.dataset, args.weights)
```

Ketika model sudah jadi, maka kita bisa menggunakan model tersebut untuk men-test data. Setelah men-load model terdapat kode untuk mem-predict image terhadap model tersebut. Kemudian untuk mengolah metric evaluasi menggunakan kode pada file. Metrics.py. dan jika ingin membuat grafik menggunakan file visualize.py.

3 HASIL, INTERPRETASI HASIL, ANALISIS

```
251/251 [==============================] - ETA: 0s - loss: 2.8066 - accuracy: 0.2625Traceback (most recent call last):
```

Setelah mengganti parameter kode

- Epoch 30 => 1
- jumlah data training >100000 data training => ±4000 data training
- jumlah data testing >10000 data training => ±400 data testing

didapat accuracy sebesar 0.26 dan loss sebesar 2.806 .

Alasan mengapa saya mengganti parameter diatas adalah karena jika saya menggunakan parameter persis seperti yg dilakukan peneliti maka akan memakan waktu berhari hari untuk mentraining data nya. Dengan spesifikasi computer jauh diatas saya peneliti melakukan training selama 44 jam. Asumsi saya menggunakan computer saya adalah > 48 jam.

Alasan mengapa saya mendapat akurasi rendah dan loss yang cukup tinggi alasannya jelas karena saya mengubah jumlah dataset yang digunakan dan jumlah epoch yang saya gunakan sangat sedikit yaitu sebanyak 1 epoch. Jumlah dataset dan jumlah epoch tentu merupakan parameter yang sangat berpengaruh didalam dunia neural network.

4 ANALISIS TAMBAHAN

Terlebih dari hasil yang sangat bagus ketika dilakukan oleh peneliti, saya sedikit bisa menyimpulkan bahwa memerlukan komputasi yang sangat besar saat melakukan training dan saat saya mencoba model terhadap single file image. jadi spesifikasi device yang digunakan harus dalam keaadan optimal. Dan juga peneliti menyarankan memperluas pekerjaan ini untuk melakukan segmentasi batas lapisan retina, sehingga fitur dan kelainan yang lebih halus dapat dideteksi secara mandiri dengan kepastian yang lebih tinggi yang dapat menjadi alat potensial bagi dokter mata di seluruh dunia.

KESIMPULAN

Pada percobaan ini dapat disimpulkan bahwa Peneliti berhasil membuat model yang mengalahkan peforma model lain. Artinya keterbaharuan teknologi yang peneliti terapkan dalam penelitiannya dapat dibilang berhasil. Terbukti bahwa residual init, building block , cnn architecture berhasil meningkatkan peforma. Dengan akurasi sebesar 99.8 pada dataset Oct2017 dan akurasi sebesar 100.00 pada dataset Srinivasan2014 membuat intervensi ahli kurang dibutuhkan dikarenakan metric evaluasi yang didapat sudah sangat bagus.

REFERENSI

[1] C. for Disease Control, Prevention et al., “National diabetes statistics report, 2017,” 2017.

[2] J. W. Yau, S. L. Rogers, R. Kawasaki, E. L. Lamoureux, J. W. Kowalski, T. Bek, S.-J. Chen, J. M. Dekker, A. Fletcher, J. Grauslund et al., “Global prevalence and major risk factors of diabetic retinopathy,” Diabetes care, vol. 35, no. 3, pp. 556–564, 2012.

[3] D. S. W. Ting, G. C. M. Cheung, and T. Y. Wong, “Diabetic retinopathy: global prevalence, major risk factors, screening practices and public health challenges: a review,” Clinical & experimental ophthalmology, vol. 44, no. 4, pp. 260–277, 2016.

[4] R. R. Bourne, G. A. Stevens, R. A. White, J. L. Smith, S. R. Flaxman, H. Price, J. B. Jonas, J. Keeffe, J. Leasher, K. Naidoo et al., “Causes of vision loss worldwide, 1990–2010: a systematic analysis,” The lancet global health, vol. 1, no. 6, pp. e339–e349, 2013.

[5] P. P. Srinivasan, L. A. Kim, P. S. Mettu, S. W. Cousins, G. M. Comer, J. A. Izatt, and S. Farsiu, “Fully automated detection of diabetic macular edema and dry age-related macular degeneration from optical coherence tomography images,” Biomedical optics express, vol. 5, no. 10, pp. 3568–3577, 2014.

[6] K. Alsaih, G. Lemaitre, M. Rastgoo, J. Massich, D. Sidibe, and F. Meri- ´audeau, “Machine learning techniques for diabetic macular edema (dme) classification on sd-oct images,” Biomedical engineering online, vol. 16, no. 1, p. 68, 2017.

[7] D. S. Friedman, B. J. OColmain, B. Munoz, S. C. Tomany, C. McCarty, P. De Jong, B. Nemesure, P. Mitchell, J. Kempen et al., “Prevalence of age-related macular degeneration in the united states,” Arch ophthalmol, vol. 122, no. 4, pp. 564–572, 2004.

[8] N. Ferrara, “Vascular endothelial growth factor and age-related macular degeneration: from basic science to therapy,” Nature medicine, vol. 16, no. 10, p. 1107, 2010.

[9] W. L. Wong, X. Su, X. Li, C. M. G. Cheung, R. Klein, C.-Y. Cheng, and T. Y. Wong, “Global prevalence of age-related macular degeneration and disease burden projection for 2020 and 2040: a systematic review and meta-analysis,” The Lancet Global Health, vol. 2, no. 2, pp. e106–e116, 2014.

[10] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan et al., “Identifying medical diagnoses and treatable diseases by image-based deep learning,” Cell, vol. 172, no. 5, pp. 1122–1131, 2018.

[11] H. Nguyen, A. Roychoudhry, and A. Shannon, “Classification of diabetic retinopathy lesions from stereoscopic fundus images,” in Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society.’Magnificent Milestones and Emerging Opportunities in Medical Engineering’(Cat. No. 97CH36136), vol. 1. IEEE, 1997, pp. 426–428.

[12] B. M. Ege, O. K. Hejlesen, O. V. Larsen, K. Møller, B. Jennings, D. Kerr, and D. A. Cavan, “Screening for diabetic retinopathy using computer based image analysis and statistical classification,” Computer methods and programs in biomedicine, vol. 62, no. 3, pp. 165–175, 2000.

[13] G. Panozzo, B. Parolini, E. Gusson, A. Mercanti, S. Pinackatt, G. Bertoldo, and S. Pignatto, “Diabetic macular edema: an oct-based classification,” in Seminars in ophthalmology, vol. 19, no. 1-2. Taylor & Francis, 2004, pp. 13–20.

[14] C. I. Sanchez, R. Hornero, M. I. Lopez, and J. Poza, “Retinal image analysis to detect and quantify lesions

associated with diabetic retinopathy," in The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, vol. 1. IEEE, 2004, pp. 1624–1627.

[15] R. A. Costa, M. Skaf, L. A. Melo Jr, D. Calucci, J. A. Cardillo, J. C. Castro, D. Huang, and M. Wojtkowski, "Retinal assessment using optical coherence tomography," Progress in retinal and eye research, vol. 25, no. 3, pp. 325–353, 2006.

[16] X. C. MeindertNiemeijer, L. Z. K. Lee, M. D. Abramoff, and M. Sonka, `"3d segmentation of fluid-associated abnormalities in retinal oct: Probability constrained graph-search-graph-cut," IEEE Transactions on Medical Imaging, vol. 31, no. 8, pp. 1521-1531, 2012.

[17] A. Lang, A. Carass, M. Hauser, E. S. Sotirchos, P. A. Calabresi, H. S. Ying, and J. L. Prince, "Retinal layer segmentation of macular oct images using boundary classification," Biomedical optics express, vol. 4, no. 7, pp. 1133–1152, 2013.

[18] A. Mishra, A. Wong, K. Bizheva, and D. A. Clausi, "Intra-retinal layer segmentation in optical coherence tomography images," Optics express, vol. 17, no. 26, pp. 23 719–23 728, 2009.

[19] G. Quellec, K. Lee, M. Dolejsi, M. K. Garvin, M. D. Abramoff, and M. Sonka, "Three-dimensional analysis of retinal layer texture: identification of fluid-filled regions in sd-oct of the macula," IEEE transactions on medical imaging, vol. 29, no. 6, pp. 1321–1330, 2010.

[20] K. Lee, M. Niemeijer, M. K. Garvin, Y. H. Kwon, M. Sonka, and M. D. Abramoff, "Segmentation of the optic disc in 3-d oct scans of the optic nerve head," IEEE transactions on medical imaging, vol. 29, no. 1, pp. 159–168, 2010.

[21] I. Ghorbel, F. Rossant, I. Bloch, S. Tick, and M. Paques, "Automated segmentation of macular layers in oct images and quantitative evaluation of performances," Pattern Recognition, vol. 44, no. 8, pp. 1590–1603, 2011.

[22] R. Kafieh, H. Rabbani, and S. Kermani, "A review of algorithms for segmentation of optical coherence tomography from retina," Journal of medical signals and sensors, vol. 3, no. 1, p. 45, 2013.

[23] J. Y. Lee, S. J. Chiu, P. P. Srinivasan, J. A. Izatt, C. A. Toth, S. Farsiu, and G. J. Jaffe, "Fully automatic software for retinal thickness in eyes with diabetic macular edema from images acquired by cirrus and spectralis systems," Investigative ophthalmology & visual science, vol. 54, no. 12, pp. 7595–7602, 2013.

[24] G. Lemaˆıtre, M. Rastgoo, J. Massich, C. Y. Cheung, T. Y. Wong, E. Lamoureux, D. Milea, F. Meriaudeau, and D. Sidib ´ e, "Classification ´ of sd-oct volumes using local binary patterns: experimental validation for dme detection," Journal of ophthalmology, vol. 2016, 2016.

[25] C. S. Lee, D. M. Baughman, and A. Y. Lee, "Deep learning is effective for classifying normal versus age-related macular degeneration oct images," Ophthalmology Retina, vol. 1, no. 4, pp. 322–327, 2017.

[26] M. Treder, J. L. Lauermann, and N. Eter, "Automated detection of exudative age-related macular degeneration in spectral domain optical coherence tomography using deep learning," Graefe's Archive for Clinical and Experimental Ophthalmology, vol. 256, no. 2, pp. 259–265, 2018.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in European conference on computer vision. Springer, 2016, pp. 630–645.

[30] L. Sifre and S. Mallat, "Rigid-motion scattering for image classification," PhD thesis, Ph. D. thesis, vol. 1, p. 3, 2014.

[31] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3156–3164.

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[34] M. Awais, H. Muller, T. B. Tang, and F. Meriaudeau, "Classification ¨ of sd-oct images using a deep learning approach," in 2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA). IEEE, 2017, pp. 489–492.

[35] S. P. K. Karri, D. Chakraborty, and J. Chatterjee, “Transfer learning based classification of optical coherence tomography images with diabetic macular edema and dry age-related macular degeneration,” Biomedical optics express, vol. 8, no. 2, pp. 579–592, 2017.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015