LAMPIRAN

Lampiran 1. Kode Arsitektur *CNN-K*

```
def build_custom_cnn(input_shape):
    inputs = Input(input_shape)
    #Encoder
    #Blok 1
    c1 = Conv2D(64, (3, 3), activation='relu',
padding='same')(inputs)
    c1 = BatchNormalization()(c1)
    c1 = Conv2D(64, (3, 3), activation='relu', padding='same')(c1)
    c1 = BatchNormalization()(c1)
    p1 = MaxPooling2D((2, 2))(c1)
    #Blok 2
    c2 = Conv2D(128, (3, 3), activation='relu',
padding='same')(p1)
    c2 = BatchNormalization()(c2)
    c2 = Conv2D(128, (3, 3), activation='relu',
padding='same')(c2)
    c2 = BatchNormalization()(c2)
    p2 = MaxPooling2D((2, 2))(c2)
    #Blok 3
    c3 = Conv2D(256, (3, 3), activation='relu',
padding='same')(p2)
    c3 = BatchNormalization()(c3)
    c3 = Conv2D(256, (3, 3), activation='relu',
padding='same')(c3)
    c3 = BatchNormalization()(c3)
    p3 = MaxPooling2D((2, 2))(c3)
    #Bottleneck
    b = Conv2D(512, (3, 3), activation='relu', padding='same')(p3)
    b = BatchNormalization()(b)
    b = Conv2D(512, (3, 3), activation='relu', padding='same')(b)
    b = BatchNormalization()(b)
    #Decoder
    #Blok 1
```

```
    u1 = Conv2DTranspose(256, (2, 2), strides=(2, 2),
padding='same')(b)

    u1 = BatchNormalization()(u1)

    u1 = concatenate([u1, c3]) # Skip connection

    c4 = Conv2D(256, (3, 3), activation='relu',
padding='same')(u1)

    c4 = BatchNormalization()(c4)

    c4 = Conv2D(256, (3, 3), activation='relu',
padding='same')(c4)

    c4 = BatchNormalization()(c4)

    #Blok 2

    u2 = Conv2DTranspose(128, (2, 2), strides=(2, 2),
padding='same')(c4)

    u2 = BatchNormalization()(u2)

    u2 = concatenate([u2, c2]) # Skip connection

    c5 = Conv2D(128, (3, 3), activation='relu',
padding='same')(u2)

    c5 = BatchNormalization()(c5)

    c5 = Conv2D(128, (3, 3), activation='relu',
padding='same')(c5)

    c5 = BatchNormalization()(c5)

    #Blok 3

    u3 = Conv2DTranspose(64, (2, 2), strides=(2, 2),
padding='same')(c5)

    u3 = BatchNormalization()(u3)

    u3 = concatenate([u3, c1]) # Skip connection

    c6 = Conv2D(64, (3, 3), activation='relu', padding='same')(u3)

    c6 = BatchNormalization()(c6)

    c6 = Conv2D(64, (3, 3), activation='relu', padding='same')(c6)

    c6 = BatchNormalization()(c6)

    #Output Layer

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c6)

    model = Model(inputs=[inputs], outputs=[outputs])

    return model
```

Lampiran 2. Kode Arsitektur *U-VGG*

```
def build_vgg16_unet(input_shape):

    vgg16 = VGG16(weights='imagenet', include_top=False,
input_shape=input_shape)

    for layer in vgg16.layers[:15]:

        layer.trainable = False

    #Encoder

    c1 = vgg16.get_layer("block1_conv2").output

    c2 = vgg16.get_layer("block2_conv2").output

    c3 = vgg16.get_layer("block3_conv3").output

    c4 = vgg16.get_layer("block4_conv3").output

    c5 = vgg16.get_layer("block5_conv3").output

    #Decoder

    u6 = UpSampling2D((2, 2))(c5); u6 = concatenate([u6, c4]); c6
= Conv2D(256, (3, 3), activation='relu', padding='same')(u6); c6 =
BatchNormalization()(c6); c6 = Dropout(0.5)(c6)

    u7 = UpSampling2D((2, 2))(c6); u7 = concatenate([u7, c3]); c7
= Conv2D(256, (3, 3), activation='relu', padding='same')(u7); c7 =
BatchNormalization()(c7); c7 = Dropout(0.5)(c7)

    u8 = UpSampling2D((2, 2))(c7); u8 = concatenate([u8, c2]); c8
= Conv2D(128, (3, 3), activation='relu', padding='same')(u8); c8 =
BatchNormalization()(c8)

    u9 = UpSampling2D((2, 2))(c8); u9 = concatenate([u9, c1]); c9
= Conv2D(64, (3, 3), activation='relu', padding='same')(u9); c9 =
BatchNormalization()(c9)

    #Output

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c9)

    return Model(inputs=vgg16.input, outputs=outputs)
```

Lampiran 3. Kode Arsitektur *DL-ResNet*

```
def ASPP(inputs):

    shape = inputs.shape

    y_1x1 = Conv2D(256, (1, 1), padding="same",
activation="relu")(inputs)
```

```
    y_3x3_r6 = Conv2D(256, (3, 3), padding="same",
activation="relu", dilation_rate=6)(inputs)

    y_3x3_r12 = Conv2D(256, (3, 3), padding="same",
activation="relu", dilation_rate=12)(inputs)

    y_3x3_r18 = Conv2D(256, (3, 3), padding="same",
activation="relu", dilation_rate=18)(inputs)

    y_pool = GlobalAveragePooling2D()(inputs)

    y_pool = tf.reshape(y_pool, [-1, 1, 1, shape[-1]])

    y_pool = Conv2D(256, 1, padding="same",
activation="relu")(y_pool)

    y_pool = UpSampling2D((shape[1], shape[2]),
interpolation="bilinear")(y_pool)

    y = concatenate([y_1x1, y_3x3_r6, y_3x3_r12, y_3x3_r18,
y_pool])

    y = Conv2D(256, (1, 1), padding="same", activation="relu")(y)

    return y


def build_deeplabv3plus(input_shape):

    #ENCODER

    backbone = ResNet50(weights="imagenet", include_top=False,
input_shape=input_shape)

    high_level_features =
backbone.get_layer("conv4_block6_out").output (output_stride=4)

    low_level_features =
backbone.get_layer("conv2_block3_out").output


    #ASPP

    x = ASPP(high_level_features)


    #DECODER

    # Upsampling output ASPP

    x = UpSampling2D(size=(4, 4), interpolation="bilinear")(x)

    low_level_features = Conv2D(48, (1, 1), padding="same",
activation="relu")(low_level_features)

    x = concatenate([x, low_level_features])

    x = Conv2D(256, (3, 3), padding="same", activation="relu")(x)
```

```
    x = Conv2D(256, (3, 3), padding="same", activation="relu")(x)

    # Final Upsampling
    x = UpSampling2D(size=(4, 4), interpolation="bilinear")(x)

    #Output Layer
    outputs = Conv2D(1, (1, 1), activation="sigmoid")(x)
    model = Model(inputs=backbone.input, outputs=outputs)
    return model
```

RIWAYAT HIDUP

Putu Haryaka Seta Dewa lahir di Busungbiu pada tanggal 12 Mei 2004. Penulis berkewarganegaraan Indonesia dan beragama Hindu. Saat ini penulis bertempat tinggal di Busungbiu, Buleleng, Bali, Indonesia. Penulis menempuh pendidikan Sekolah Dasar di SD Negeri 5 Sanur dan lulus pada jenjang tersebut. Selanjutnya, penulis melanjutkan pendidikan di SMP Negeri 4 Busungbiu dan lulus pada jenjang tersebut. Penulis kemudian menempuh pendidikan menengah atas di SMA Negeri 1 Busungbiu dengan penjurusan Bahasa dan Sastra hingga menyelesaikan pendidikan. Penulis terdaftar sebagai mahasiswa Program Studi S1 Ilmu Komputer Universitas Pendidikan Ganesha sejak tahun 2022 sampai dengan penyusunan skripsi ini.