Image Segmentation

DeepLabv3+ implementation lengkap untuk semantic segmentation. Chapter ini fokus pada pixel-wise prediction, atrous convolutions, dan production tf.data pipelines untuk large-scale image segmentation. [1]

Dataset Analysis dan Preprocessing

Cityscapes/Pascal VOC EDA:

- 19 semantic classes (person, car, road, etc.) •

Class imbalance: background >> foreground •

High resolution images 2048 1024

Preprocessing Pipeline:

```
def preprocess_image(image, mask):
    # Resize dengan aspect ratio preservation
    h, w = 512, 1024
    image = tf.image.resize(image, (h, w), method='bilinear')
    mask = tf.image.resize(mask, (h, w), method='nearest')

    # Normalize
    image = image / 255.0
    return image, mask
```

Optimized tf.data Pipelines untuk Segmentation

```
def get_segmentation_pipeline(image_paths, mask_paths, batch_size=8):
    dataset = tf.data.Dataset.from_tensor_slices((image_paths, mask_paths))
    dataset = dataset.map(preprocess_image, num_parallel_calls=AUTOTUNE)

    # Cache + shuffle + repeat + batch + prefetch
    dataset = dataset.cache()
    dataset = dataset.shuffle(1000)
    dataset = dataset.repeat()
    dataset = dataset.batch(batch_size)
    dataset = dataset.prefetch(AUTOTUNE)
    return dataset
```

DeepLabv3   Architecture Detail

Backbone: ResNet50   Atrous Spatial Pyramid Pooling ASPP **Atrous**

Convolution Dilated Convolution):

```
def atrous_conv(filters, rate):
    return tf.keras.layers.Conv2D(
        filters, 3, padding='same',
        dilation_rate=rate,
        use_bias=False
    )
```

ASPP Module Multi-scale context):

```
def aspp(inputs):
    # Global average pooling branch
    shape = tf.shape(inputs)
    b0 = GlobalAveragePooling2D()(inputs)
    b0 = tf.keras.layers.Reshape((1, 1, 256))(b0)
    b0 = Conv2D(256, 1, use_bias=False)(b0)
    b0 = BilinearUpsample((shape[^1], shape[^2]))(b0)

    # 1x1 conv
    b1 = Conv2D(256, 1, use_bias=False)(inputs)

    # Atrous convs (rates 6, 12, 18)
    b2 = atrous_conv(256, 6)(inputs)
    b3 = atrous_conv(256, 12)(inputs)
    b4 = atrous_conv(256, 18)(inputs)

    return Concatenate()([b0, b1, b2, b3, b4])
```

Decoder (refine coarse output):

```
def decoder(aspp_out, low_level_features):
    aspp_out = Conv2D(256, 3, padding='same')(aspp_out)
    low_level = Conv2D(48, 1, padding='same')(low_level_features)

    x = Concatenate()([aspp_out, BilinearUpsample()(low_level)])
    x = Conv2D(num_classes, 1, activation='softmax')
    return BilinearUpsample((512, 1024))(x)
```

Loss Functions dan Evaluation Metrics

Hybrid Loss: Weighted CrossEntropy + Dice Loss

```
def dice_loss(y_true, y_pred, smooth=1):
    y_true_f = tf.keras.backend.flatten(y_true)
    y_pred_f = tf.keras.backend.flatten(y_pred)
    intersection = tf.keras.backend.sum(y_true_f * y_pred_f)
    return 1 - (2. * intersection + smooth) / (tf.keras.backend.sum(y_true_f) + tf.keras.

def total_loss(y_true, y_pred):
    return tf.keras.losses.categorical_crossentropy(y_true, y_pred) + dice_loss(y_true, y
```

Metrics: mIoU (mean Intersection over Union), per-class IoU.

Kesimpulan

DeepLabv3+ dengan atrous convolutions dan ASPP adalah state-of-the-art untuk semantic segmentation tasks. [1]

```
text_vectorizer = TextVectorization(
    max_tokens=10000,
    output_sequence_length=300,  # 95th percentile
    standardize='lower_and_strip_punctuation',
    output_mode='int'
)
text_vectorizer.adapt(train_texts)
```