Image Classification with CNNs

Implementasi lengkap InceptionNet v1 GoogLeNet) untuk 200-class image classification. Chapter ini mencakup EDA komprehensif, production data pipelines, dan state-of-the-art CNN architecture details. [1]

Exploratory Data Analysis EDA Mendalam

Dataset Analysis Pipeline:

```
def analyze_dataset(image_paths, labels):
    heights, widths = [], []
    class_dist = {}

    for path, label in zip(image_paths, labels):
        img = tf.io.read_file(path)
        img = tf.io.decode_image(img, channels=3)
        heights.append(img.shape[^0])
        widths.append(img.shape[^1])
        class_dist[label] += 1

    plt.figure(figsize=(15, 5))
    plt.subplot(131); plt.hist(heights); plt.title('Height Distribution')
    plt.subplot(132); plt.hist(widths); plt.title('Width Distribution')
    plt.subplot(133); plt.bar(class_dist.keys(), class_dist.values())
```

Key insights: Class imbalance, resolution variance, color channel statistics.

Production Image Data Pipelines

Keras ImageDataGenerator + tf.data Hybrid:

```
# Legacy ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale=1./255,
```

```
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

train_generator = train_datagen.flow_from_directory(
    'data/train', target_size=(64, 64), batch_size=32, class_mode='categorical'
)

# Modern tf.data approach
def augment(image, label):
    image = tf.image.random_flip_left_right(image)
    image = tf.image.random_brightness(image, 0.2)
    image = tf.image.random_contrast(image, 0.8, 1.2)
    return image, label

train_ds = image_ds.map(augment, num_parallel_calls=AUTOTUNE)
```

InceptionNet v1 Complete Implementation

Stem Network (feature extraction):

```
def stem(inputs):
    x = Conv2D(64, 7, strides=2, padding='same')(inputs)
    x = MaxPool2D(3, strides=2, padding='same')(x)
    x = Conv2D(64, 1)(x)
    x = Conv2D(192, 3, padding='same')(x)
    x = MaxPool2D(3, strides=2, padding='same')(x)
    return x
```

Inception Block (multi-scale feature extraction):

```
def inception_block(prev_layer, nums_filters):
    # 1x1 conv
    conv_1x1 = Conv2D(nums_filters[^0], 1, padding='same', activation='relu')(prev_layer)

    # 1x1 -> 3x3
    conv_3x3 = Conv2D(nums_filters[^1], 1, padding='same', activation='relu')(prev_layer)
    conv_3x3 = Conv2D(nums_filters[^2], 3, padding='same', activation='relu')(conv_3x3)

    # 1x1 -> 5x5
    conv_5x5 = Conv2D(nums_filters[^3], 1, padding='same', activation='relu')(prev_layer)
    conv_5x5 = Conv2D(nums_filters[^4], 5, padding='same', activation='relu')(conv_5x5)

    # Pool -> 1x1
    pool = MaxPool2D(3, strides=1, padding='same')(prev_layer)
    pool = Conv2D(nums_filters[^5], 1, padding='same', activation='relu')(pool)

    outputs = Concatenate(axis=-1)([conv_1x1, conv_3x3, conv_5x5, pool])
    return outputs
```

Auxiliary Classifiers (gradient stabilization):

```
def aux_classifier(inputs):
    x = AvgPool2D(5, strides=3)(inputs)
    x = Conv2D(128, 1, activation='relu')(x)
    x = Flatten()(x)
    x = Dense(1024, activation='relu')(x)
    x = Dense(200, activation='softmax', name='aux_output')(x)
    return x
```

Complete Model 9 Inception blocks + 2 auxiliary outputs):

```
def inception_v1():
    inputs = Input(shape=(64, 64, 3))

    # Stem
    x = stem(inputs)

    # Inception blocks dengan auxiliary classifiers
    x1 = inception_block(x, [64, 96, 128, 16, 32, 32])  # 3a
    x2 = inception_block(x1, [128, 128, 192, 32, 96, 64])  # 3b
    # ... (7 more blocks)

    # Final classifier
    x = GlobalAveragePooling2D()(x)
    predictions = Dense(200, activation='softmax')(x)

    model = Model(inputs, [predictions, aux1, aux2])
    model.compile(
        optimizer='adam',
        loss={'main_output': 'categorical_crossentropy', 'aux1': 'categorical_crossentrop
        loss_weights={'main_output': 1.0, 'aux1': 0.3, 'aux2': 0.3}
    )
    return model
```

Training dan Evaluation Strategy

Multi-output loss weighting: Main output 1.0 , auxiliary 0.3 each) untuk gradient flow yang stabil.

Learning rate scheduling dan early stopping untuk optimal convergence.

Kesimpulan

InceptionNet v1 memperkenalkan multi-scale processing dan auxiliary classifiers yang menjadi foundation modern CNN architectures. [1]