M09 - Diskusi.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

Files

..
data
cats
dogs
logs
sample_data
cat_test.jpg
dog_test2.jpg

Disk 80.15 GB available

+ Code + Text

T4 RAM Disk Gemini

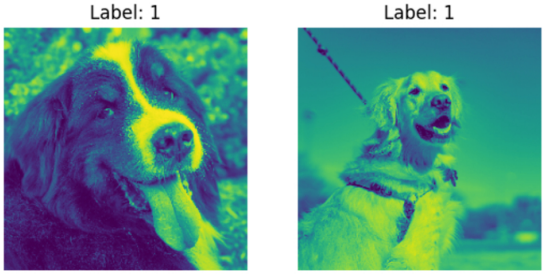
[1]
```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import TensorBoard
import os
import cv2
```

[2]
```
# Load dataset
data_dir = '/content/data'
data = tf.keras.utils.image_dataset_from_directory(data_dir, image_size=(256, 256), batch_size=2, color_mode="grayscale")
```

```
Found 20 files belonging to 2 classes.
```

[3]
```
import matplotlib.pyplot as plt

# Iterasi dataset
for images, labels in data.take(1):  # Ambil satu batch pertama
    print(f'Images shape: {images.shape}')
    print(f'Labels: {labels.numpy()}')

    # Tampilkan beberapa gambar
    plt.figure(figsize=(10, 10))
    for i in range(2):  # Tampilkan 2 gambar pertama
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(f'Label: {labels[i].numpy()}')
        plt.axis("off")
    plt.show()
```

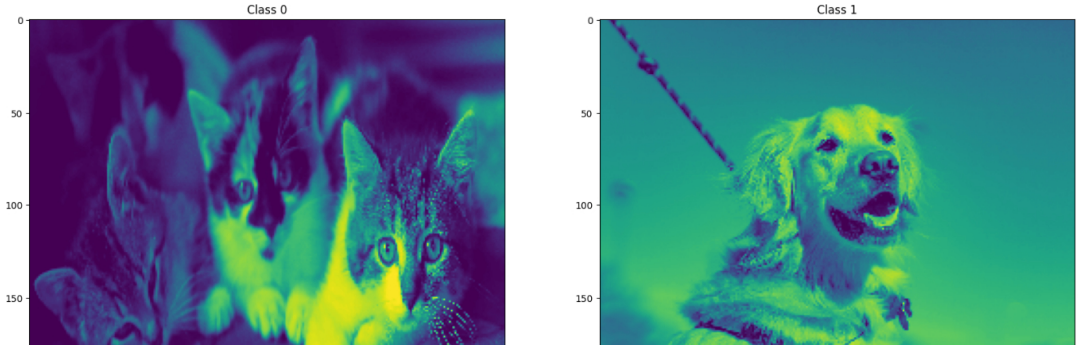
```
Images shape: (2, 256, 256, 1)
Labels: [1 1]
```

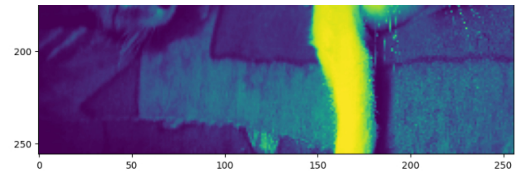


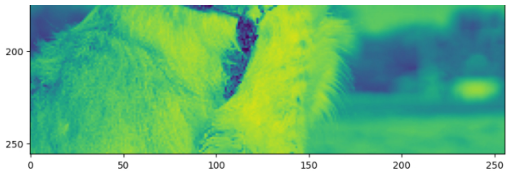

```
# Split dataset
train_size = int(0.7 * len(data))
val_size = int(0.2 * len(data))
test_size = int(0.1 * len(data))

print(train_size, val_size, test_size)

train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size + val_size).take(test_size)
```

```
7 2 1
```

[8]
```
# Visualize the data
data_iterator = train.as_numpy_iterator()
batch = data_iterator.next()

fig, ax = plt.subplots(ncols=2, figsize=(20, 20))
for idx, img in enumerate(batch[0][:2]):
    ax[idx].imshow(img.astype('uint8'))
    ax[idx].title.set_text('Class ' + str(batch[1][idx]))
plt.show()
```







[31]
```python
# Build the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(256, 256, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),  # Prevent overfitting
    Dense(2, activation='softmax')  # Binary classification
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

[32]
```python
# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```

[33]
```python
# Display model summary
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 256, 256, 32)	320
max_pooling2d_3 (MaxPooling2D)	(None, 128, 128, 32)	0
conv2d_4 (Conv2D)	(None, 128, 128, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 64, 64, 64)	0
conv2d_5 (Conv2D)	(None, 64, 64, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 32, 32, 128)	0
flatten_1 (Flatten)	(None, 131072)	0
dense_2 (Dense)	(None, 256)	33,554,688
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 2)	514

Total params: 33,647,874 (128.36 MB)
Trainable params: 33,647,874 (128.36 MB)
Non-trainable params: 0 (0.00 B)

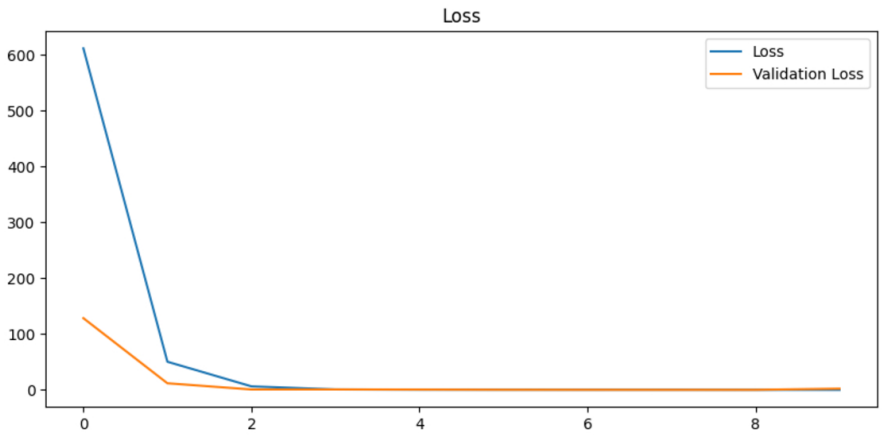
[34]
```python
# Set up TensorBoard
logdir = 'logs'
tensorboard_callback = TensorBoard(log_dir=logdir)
```

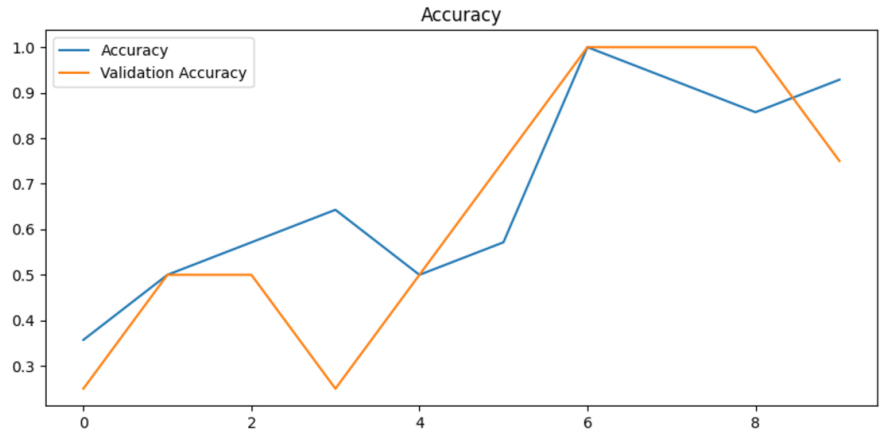
[35]
```python
# Train the model
hist = model.fit(train, validation_data=val, epochs=10, callbacks=[tensorboard_callback])
```

```
Epoch 1/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 5s 242ms/step - accuracy: 0.3507 - loss: 792.6967 - val_accuracy: 0.2500 - val_loss: 128.5207
Epoch 2/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.2792 - loss: 73.3909 - val_accuracy: 0.5000 - val_loss: 11.9977
Epoch 3/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 78ms/step - accuracy: 0.5564 - loss: 11.6211 - val_accuracy: 0.5000 - val_loss: 0.9012
Epoch 4/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5326 - loss: 1.2276 - val_accuracy: 0.2500 - val_loss: 0.9697
Epoch 5/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - accuracy: 0.5208 - loss: 0.5553 - val_accuracy: 0.5000 - val_loss: 0.7157
Epoch 6/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 81ms/step - accuracy: 0.5960 - loss: 0.5014 - val_accuracy: 0.7500 - val_loss: 0.4101
Epoch 7/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.4331 - val_accuracy: 1.0000 - val_loss: 0.3219
Epoch 8/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.9592 - loss: 0.4484 - val_accuracy: 1.0000 - val_loss: 0.2669
Epoch 9/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.8143 - loss: 0.3856 - val_accuracy: 1.0000 - val_loss: 0.2754
Epoch 10/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - accuracy: 0.8290 - loss: 0.2768 - val_accuracy: 0.7500 - val_loss: 2.3502
```

[36]
```python
# Plot loss and accuracy
fig, ax = plt.subplots(2, 1, figsize=(10, 10))
ax[0].plot(hist.history['loss'], label='Loss')
ax[0].plot(hist.history['val_loss'], label='Validation Loss')
ax[0].legend()
ax[0].set_title('Loss')

ax[1].plot(hist.history['accuracy'], label='Accuracy')
ax[1].plot(hist.history['val_accuracy'], label='Validation Accuracy')
```

```
ax[1].legend()
ax[1].set_title('Accuracy')
plt.show()
```

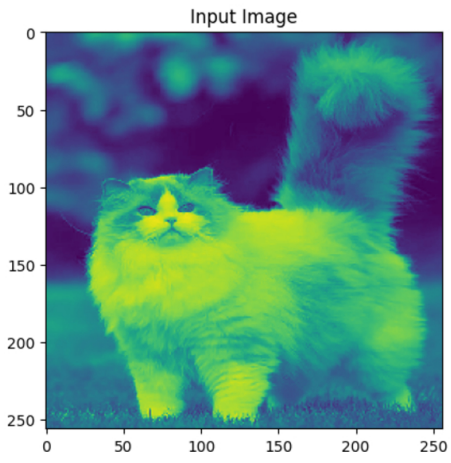





```
[13] # Evaluate model on the test set
     test_metrics = model.evaluate(test)
     print(f"Test Loss: {test_metrics[0]}, Test Accuracy: {test_metrics[1]}")
```

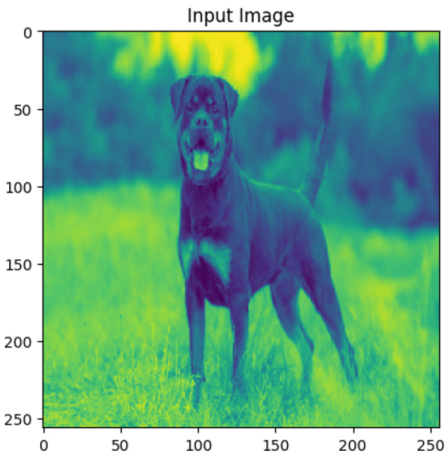
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 196ms/step - accuracy: 1.0000 - loss: 6.9737e-06
Test Loss: 6.973694780754158e-06, Test Accuracy: 1.0
```

```
# Test the model with a sample image
test_img = tf.keras.utils.load_img("/content/cat_test.jpg", target_size=(256, 256))  # Resize to match model input size
test_img = tf.image.rgb_to_grayscale(test_img)
test_img = tf.keras.utils.img_to_array(test_img)
test_label = 0

plt.imshow(test_img.astype('uint8'))
plt.title('Input Image')
plt.show()

yhat = model.predict(np.expand_dims(test_img, axis=0))
yidx = np.argmax(yhat)

if yidx == 1:
  print("Dog")
else:
  print("Cat")
print(yhat, test_label)
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 277ms/step
Cat
[[9.9967635e-01 3.2370334e-04]] 0
```

[38]
```
# Test the model with a sample image
test_img = tf.keras.utils.load_img("/content/dog_test2.jpg", target_size=(256, 256))  # Resize to match model input size
test_img = tf.image.rgb_to_grayscale(test_img)
test_img = tf.keras.utils.img_to_array(test_img)
test_label = 1

plt.imshow(test_img.astype('uint8'))
plt.title('Input Image')
plt.show()

yhat = model.predict(np.expand_dims(test_img, axis=0))
yidx = np.argmax(yhat)

if yidx == 1:
  print("Dog")
else:
  print("Cat")
print(yhat, test_label)
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
Cat
[[0.73023164 0.26976836]] 1
```

Start coding or generate with AI.