Specific Test II. Optical Character Recognition bold text

Extracted Ground Truth

```
ground_truth = [
    "Al INFINITAMENTE AMABLE NIÑO JESUS. A Vos, Dulcissimo Niño JESUS, que no solo os dignas
    "guro disseño de su edad: la Reli- gion para con Dios en la devota assistencia a los Tem
    "crianza de la niñez. Assi sea, Divinissimo Niño, por vuestra gracia, assi sea, a vuestr
]
```

PaddleOCR is an open-source Optical Character Recognition (OCR) system based on a combination of Convolutional, Recurrent, and Transformer architectures.

> Reasons to choose PaddleOCR,
>
> - It supports multi-language OCR, including Latin and non-Latin scripts (Chinese, Arabic, etc.).
> - It provides fast inference and lightweight models for real-time applications.
> - It supports spanish language
> - It integrates deep learning-based models that outperform traditional OCR tools like Tesseract in accuracy and adaptability.

```
pip install jiwer
```

Show hidden output

```
!pip install paddlepaddle paddleocr
```

Show hidden output

Import Libraries and Load Test Image

```
from paddleocr import PaddleOCR
from jiwer import wer, cer

image_path = "pdf2_page-0001.jpg"
```

Initialize Model and extract Ground Truth

```
# Initialize OCR model (Spanish)
ocr = PaddleOCR(use_angle_cls=True, lang='es', split_mode=True)

ground_truth_text = "Al INFINITAMENTE AMABLE NIÑO JESUS. A Vos, Dulcissimo Niño JESUS, que r
```

```
[2025/03/14 19:16:00] ppocr DEBUG: Namespace(help='==SUPPRESS==', use_gpu=False, use_xpu
```

Perform OCR and Extract OCR text output

```
result = ocr.ocr(image_path, cls=True)
recognized_text = " ".join([line[1][0] for line in result[0]])  # Concatenate detected text
```

```
[2025/03/14 19:16:26] ppocr DEBUG: dt_boxes num : 26, elapsed : 0.3174426555633545
[2025/03/14 19:16:26] ppocr DEBUG: cls num  : 26, elapsed : 0.10707879066467285
[2025/03/14 19:16:28] ppocr DEBUG: rec_res num  : 26, elapsed : 2.1962740421295166
```

Evaluation Metrics

CER measures the number of character-level errors (insertions, deletions, substitutions) in the recognized text compared to the ground truth.

CER = (S+D+I) / N

S = Number of substitutions (wrong character instead of correct one)

D = Number of deletions (missing character)

I = Number of insertions (extra character)

N = Total number of characters in the ground truth

WER is similar to CER but measures errors at the word level.

WER = (S+D+I) / N

S = Word substitutions

D = Word deletions

I = Word insertions

N = Total number of words in the ground truth

```
# Compute CER and WER
cer_value = cer(ground_truth_text, recognized_text)
wer_value = wer(ground_truth_text, recognized_text)

print(f"Recognized Text: {recognized_text}")
```

```
print(f"CER: {cer_value:.4f}")
print(f"WER: {wer_value:.4f}")
```

```
Recognized Text: X x X AT 'AMABLE INFINITAMENTE NINO JESUS Vos , Dulcifsimo Nino .JEsus
CER: 0.1951
WER: 0.5575
```

Other experimented OCR techniques

TESSERACT - CRNN architecture, CTC Loss

```
# Install Tesseract OCR
!sudo apt install tesseract-ocr

# Install the Spanish language data for Tesseract
!sudo apt install tesseract-ocr-spa

# Install pytesseract and OpenCV
!pip install pytesseract opencv-python #pytesseract - handle tesseract using python
```

Show hidden output

Import Libraries and Load test image

```
import cv2
import pytesseract

# Load the image
image_path = "pdf2_page-0001.jpg"
image = cv2.imread(image_path)
```

Convert the image to grayscale (improves OCR accuracy)

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Perform OCR to detect Spanish text

```
text = pytesseract.image_to_string(gray, lang='spa')

# Print the detected text
```

```
print("Detected Spanish Text:")
print(text)
```

```
Detected Spanish Text:
XAO

A
INFINITAM “AMABLE
' NINO TESUS.

e] Vos , Dulcifsimo Niño
MG ANDA] Jesus , queno lolo OS Ex!/ai.53:
| dignafteis de llamaros :8-

Dottor de los Niños, £* 4":
fino tambien de aísif *

tir como Niño entre los Doctores,

le confagra humilde efta pequeña
Inftruccion de los Niños. Es aísi,

que ella tambien fe dirige a laju-

ventud ; pero aelta, como recuer-
do de lo que aprendió, alos Nis

ños , como precifa explicacion de

lo que deben eftudiar. Por efté fo-
logritulo:es muy vueftra ¿ «y por

fer para Niños , que confiais á la
educacion de vueftra Compañía,

lo es mucho mas. En Vos, ( Divi-

no Exemplar de todas las virtus

des ) tienen abreviado el mas (e-

. g 2 gura
```

EASY OCR - VGG and LSTM based architectures, CTC(loss)

```
pip install easyocr
```

Show hidden output

```
import easyocr

# Initialize EasyOCR reader with Spanish language
reader = easyocr.Reader(['es'])

# Perform OCR on the image
result = reader.readtext('pdf2_page-0004.jpg')

for detection in result:
    print(f"Detected text: {detection[1]} (confidence: {detection[2]})")
```

Show hidden output

Specific Test I. Layout Organization Recognition

1st Technique

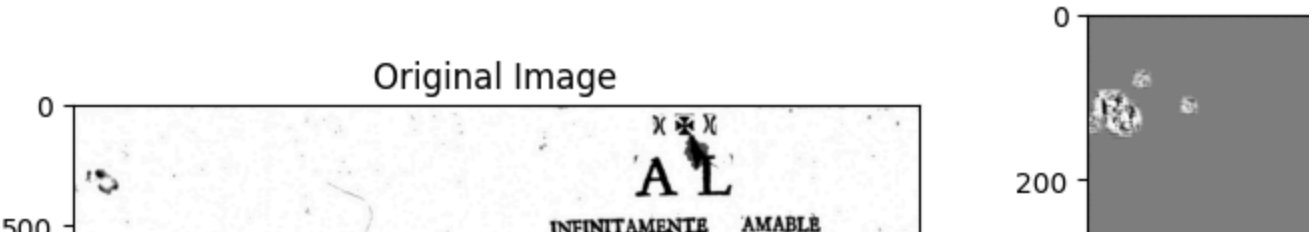
1. Install necessary libraries: TensorFlow, OpenCV, and Matplotlib for image processing and model training.
2. Load and preprocess the image: Convert to grayscale, resize to 1024×1024, and apply binary thresholding to enhance text visibility.
3. Build a simple U-Net model for layout detection: Consists of an encoder (feature extraction using CNN layers), a bottleneck, and a decoder (upsampling for segmentation).
4. Detect layout: Preprocess the image, pass it through the model, and generate a segmentation mask predicting text regions.
5. Visualize results: Display the original image alongside the predicted segmentation mask for analysis.

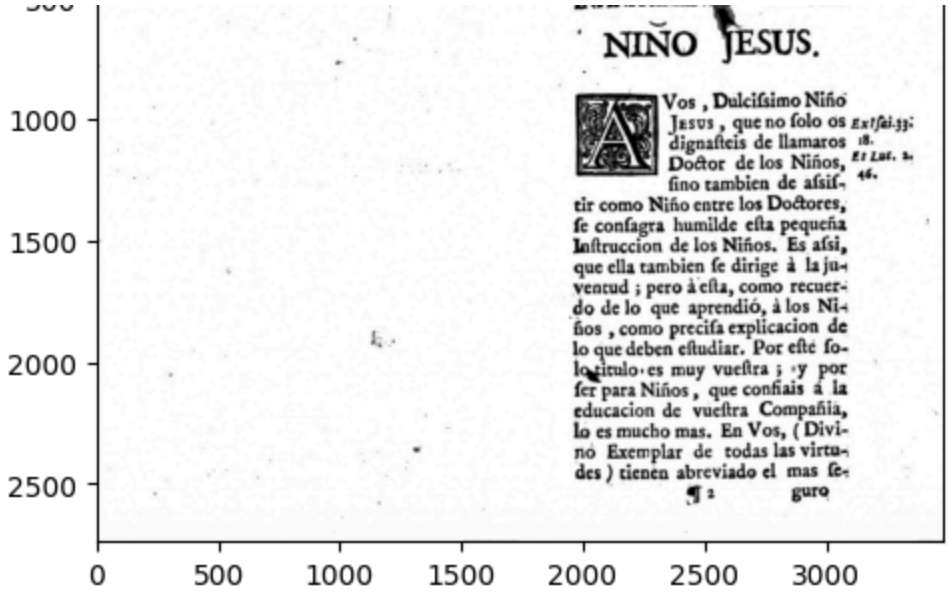
```
!pip install tensorflow opencv-python matplotlib

import cv2
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

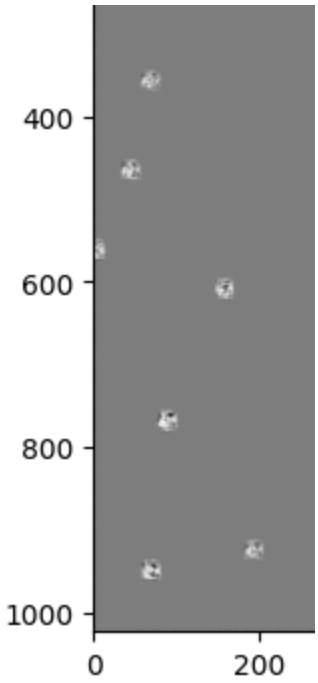
```
import matplotlib.pyplot as plt

# Load and preprocess the image
def preprocess_image(image_path):
    # Read the image in grayscale
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    img = cv2.resize(img, (1024, 1024))
    # Apply binary thresholding (can be adjusted as per the manuscript quality)
    _, img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)
    return img

# Layout detection model: simple U-Net for semantic segmentation
def build_unet_model(input_shape=(1024, 1024, 1)):
    inputs = layers.Input(shape=input_shape)

    # Encoder
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(inputs)
    pool1 = layers.MaxPooling2D(2)(conv1)

    conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(pool1)
    pool2 = layers.MaxPooling2D(2)(conv2)

    # Bottleneck
    conv3 = layers.Conv2D(256, 3, activation='relu', padding='same')(pool2)

    # Decoder
    up1 = layers.UpSampling2D(2)(conv3)
    conv4 = layers.Conv2D(128, 3, activation='relu', padding='same')(up1)

    up2 = layers.UpSampling2D(2)(conv4)
    conv5 = layers.Conv2D(64, 3, activation='relu', padding='same')(up2)

    # Output layer (segmentation map)
    output = layers.Conv2D(1, 1, activation='sigmoid')(conv5)

    model = keras.Model(inputs, output)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model

# Detect layout using trained model
def detect_layout(model, image_path):
    img = preprocess_image(image_path)
    img = np.expand_dims(img, axis=-1)  # Add channel dimension for grayscale
    img = np.expand_dims(img, axis=0)   # Add batch dimension

    # Get the prediction (segmentation mask)
    prediction = model.predict(img)
    prediction = prediction.squeeze()  # Remove batch and channel dimension

    return prediction
```

```
# Visualize the layout detection result
def visualize_result(image_path, prediction):
    img = cv2.imread(image_path)
    plt.figure(figsize=(12, 12))
    plt.subplot(1, 2, 1)
    plt.imshow(img)
    plt.title('Original Image')

    plt.subplot(1, 2, 2)
    plt.imshow(prediction, cmap='gray')
    plt.title('Detected Layout')
    plt.show()

# Example usage
model = build_unet_model()

# Path to the manuscript image
image_path = 'pdf2_page-0001.jpg'

# Detect the layout
pred = detect_layout(model, image_path)

# Visualize the result
visualize_result(image_path, pred)
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/pyt
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dis
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/py
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from ker
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from ke
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from k
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dis
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/p
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (fr
1/1 ━━━━━━━━━━━━━━━━━━━━ 9s 9s/step
```







2nd Technique

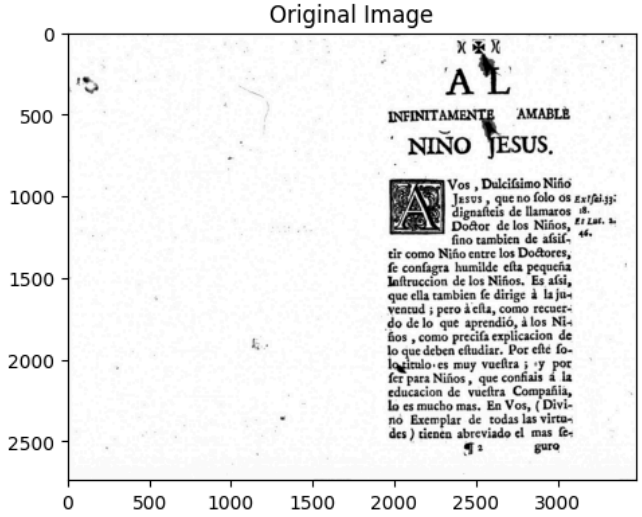
1. Load and preprocess the image: Convert to grayscale and resize it to 1024×1024 for consistency.
2. Detect tightly packed areas: Use Canny edge detection to find contours, filter based on area, and apply morphological operations to highlight dense regions.
3. Read text using Tesseract OCR: Extract text from the detected tightly packed areas using pytesseract.image_to_string()
4. Visualize results: Display the original image and the detected packed areas side by side for comparison.

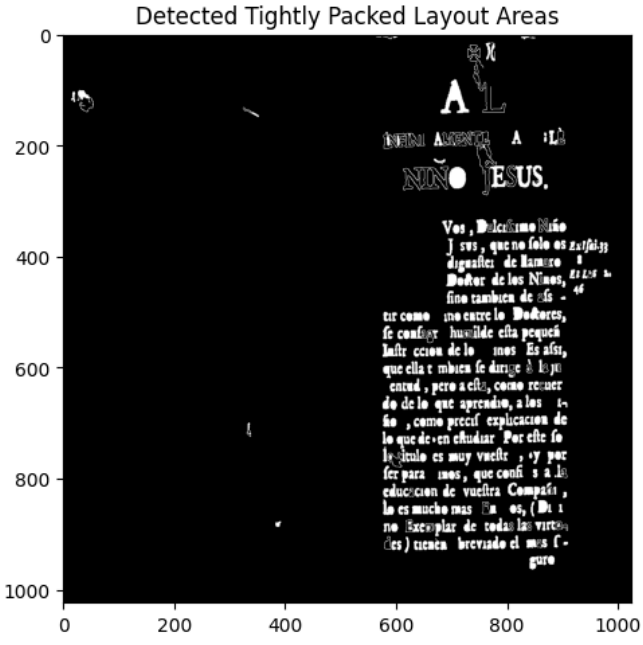
```
import cv2
import pytesseract
import numpy as np
import matplotlib.pyplot as plt

# Load and preprocess the image
def preprocess_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (1024, 1024))
    return img

# Detect tightly packed areas based on Canny edge detection and contour filtering
def detect_tightly_packed_areas(img):
```

```
        edges = cv2.Canny(img, threshold1=50, threshold2=150)
        contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        filtered_contours = []
        for contour in contours:
            area = cv2.contourArea(contour)
            if area > 30 and area < 10000:
                filtered_contours.append(contour)

        kernel = np.ones((5, 5), np.uint8)
        eroded_edges = cv2.erode(edges, kernel, iterations=1)

        packed_areas = np.zeros_like(img)
        cv2.drawContours(packed_areas, filtered_contours, -1, 255, thickness=cv2.FILLED)

        return packed_areas

    # Function to read text from image using Tesseract
    def read_text_from_image(image):
        # Convert the binary image to the proper format for Tesseract
        text = pytesseract.image_to_string(image, config='--psm 6')
        return text

    # Visualize and read text from the tightly packed areas
    def visualize_and_read_text(image_path, packed_areas):
        # Read text from the packed_areas
        detected_text = read_text_from_image(packed_areas)

        # Print the detected text
        print("Detected Text:")
        print(detected_text)

        # Read the original image for visualization
        img = cv2.imread(image_path)

        plt.figure(figsize=(12, 12))

        # Original Image
        plt.subplot(1, 2, 1)
        plt.imshow(img)
        plt.title('Original Image')

        # Detected Tightly Packed Areas
        plt.subplot(1, 2, 2)
        plt.imshow(packed_areas, cmap='gray')
        plt.title('Detected Tightly Packed Layout Areas')

        plt.show()

    image_path = 'pdf2_page-0001.jpg'
```

```
    # Load and preprocess the image
    img = preprocess_image(image_path)

    # Detect tightly packed areas
    packed_areas = detect_tightly_packed_areas(img)

    # Visualize and read text from the tightly packed areas
    visualize_and_read_text(image_path, packed_areas)
```

```
Detected Text:
yi
ie . A
NEN mast A iL
NI a) SS
Ves , Bolcrcime Niko
J sus, quene fole es axifais
Cw
Boter de les Nines, a »
fine tambien de fs - *
trcome inecatrcle Dekeres,
fe confer husilde efta pequei
Aes Loe ron
Pra Coe Ne em ay
COG Bey
ORO ry eg)

r fie ,come precif explicacien de
cr rar rae ee
lecitule es muy vacltr , sy par
Ca Oe Tae
educacien de vueltra Compain ,
le esmuche mas Fa es, (Bi

. no Sxemplar de tedas las virte
Ces) tienen breviade el mes [-
ro
```





Layout and Character Detection Combined

Import, Load Test Image and perform OCR using *PaddleOCR*

```
import cv2
import numpy as np
from paddleocr import PaddleOCR
from google.colab.patches import cv2_imshow  # Import for displaying images in Colab

# Initialize PaddleOCR (Spanish language model)
ocr = PaddleOCR(use_angle_cls=True, lang='es', split_mode=True)

# Load image path
image_path = "pdf2_page-0001.jpg"

# Perform OCR
result = ocr.ocr(image_path, cls=True)

# Read the image
image = cv2.imread(image_path)
```

```
[2025/03/14 19:41:45] ppocr DEBUG: Namespace(help='==SUPPRESS==', use_gpu=False, use_xpu
[2025/03/14 19:41:47] ppocr DEBUG: dt_boxes num : 26, elapsed : 0.3767983913421631
[2025/03/14 19:41:47] ppocr DEBUG: cls num  : 26, elapsed : 0.1016397476196289
[2025/03/14 19:41:48] ppocr DEBUG: rec_res num  : 26, elapsed : 1.3583807945251465
```

Loop through detected text and draw bounding boxes

```
for line in result[0]:
    points = line[0]  # Bounding box coordinates
    text = line[1][0]  # Recognized text

    # Convert points to integers
    points = [(int(x), int(y)) for x, y in points]

    # Draw bounding box
    cv2.polylines(image, [np.array(points)], isClosed=True, color=(0, 255, 0), thickness=2

    # Put text near the bounding box with a larger font size
    font_scale = 1.2  # Increase the font size
    font_thickness = 3  # Increase the thickness of text
    cv2.putText(image, text, (points[0][0], points[0][1] - 10),
```

Display Image

```
# Display the image in Colab
cv2_imshow(image)
```