In [4]:
```
import torch

if torch.cuda.is_available():
    device = torch.device('cuda')
    print('GPU is available. PyTorch will use the GPU.')
    print('Memory Usage:')
    print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
    print('Cached: ', round(torch.cuda.memory_reserved(0)/1024**3,1), 'GB')
else:
    device = torch.device('cpu')
    print('GPU is not available. PyTorch will use the CPU.')
```

```
GPU is available. PyTorch will use the GPU.
Memory Usage:
Allocated: 0.0 GB
Cached:  0.0 GB
```

In [5]:
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
from PIL import Image
import json
import fitz
from tqdm.notebook import tqdm
import docx
import re
import albumentations as A
from albumentations.pytorch import ToTensorV2
import segmentation_models_pytorch as smp
from sklearn.metrics import precision_score, recall_score, f1_score, jaccard_score
from transformers import TrOCRProcessor, VisionEncoderDecoderModel
import warnings
warnings.filterwarnings('ignore')

base_dir = "./"
os.makedirs(os.path.join(base_dir, "output"), exist_ok=True)
os.makedirs(os.path.join(base_dir, "output/images"), exist_ok=True)
os.makedirs(os.path.join(base_dir, "output/transcriptions"), exist_ok=True)
os.makedirs(os.path.join(base_dir, "output/masks"), exist_ok=True)
os.makedirs(os.path.join(base_dir, "output/layout_model"), exist_ok=True)
os.makedirs(os.path.join(base_dir, "output/regions"), exist_ok=True)
os.makedirs(os.path.join(base_dir, "output/ocr_model"), exist_ok=True)

pdf_dir = "Test sources"
transcription_dir = "Test transcriptions"
image_dir = os.path.join(base_dir, "output/images")
transcriptions_output_dir = os.path.join(base_dir, "output/transcriptions")
masks_dir = os.path.join(base_dir, "output/masks")
layout_model_dir = os.path.join(base_dir, "output/layout_model")
regions_dir = os.path.join(base_dir, "output/regions")
ocr_model_dir = os.path.join(base_dir, "output/ocr_model")

print("RenAIssance OCR Pipeline Configuration:")
print(f"PDF Directory: {pdf_dir}")
print(f"Transcription Directory: {transcription_dir}")
print(f"Output Directory: {base_dir}/output")
```

```
RenAIssance OCR Pipeline Configuration:
PDF Directory: Test sources
Transcription Directory: Test transcriptions
Output Directory: .//output
```

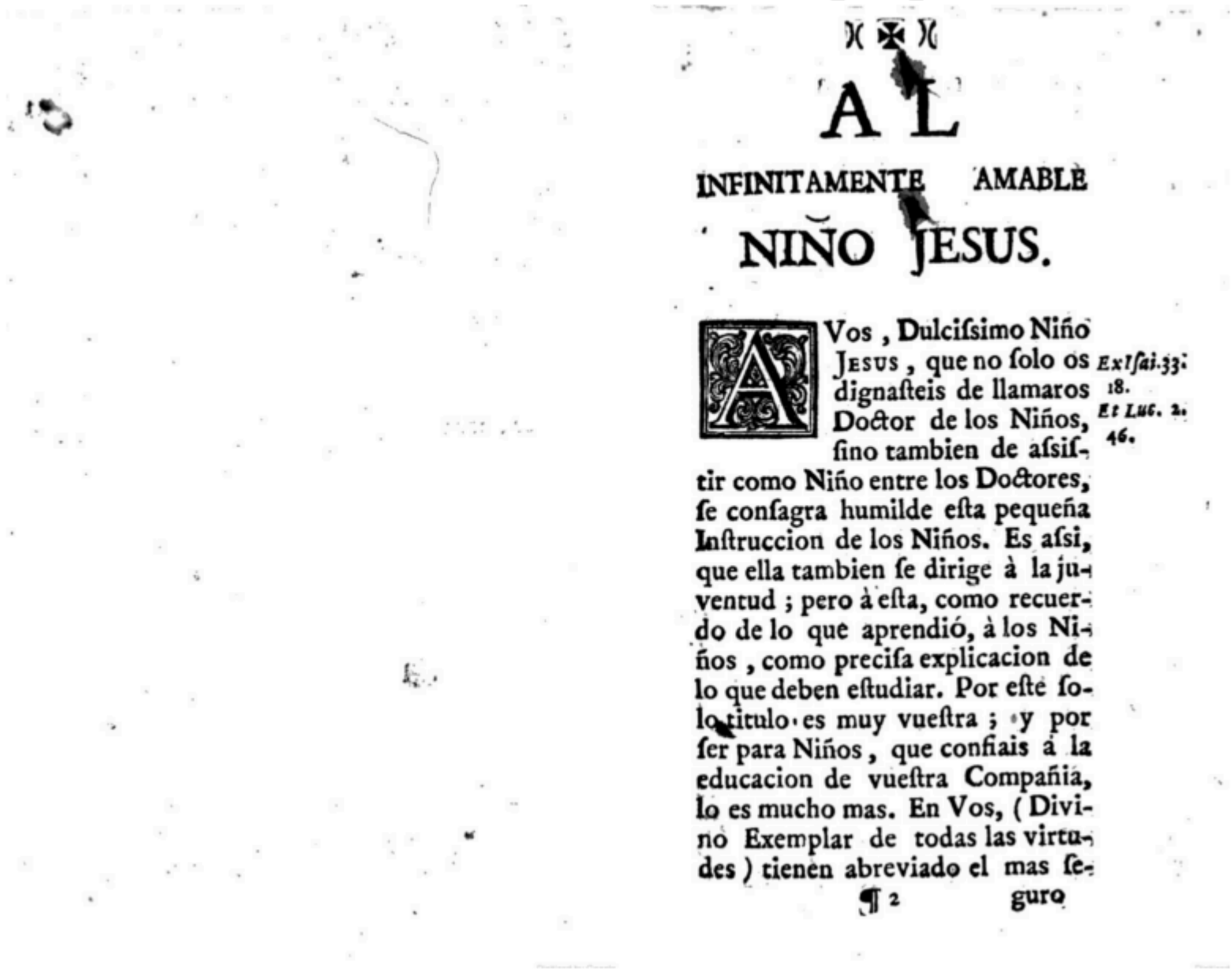
In [6]:
```
# 1. PDF to Images Conversion

Image.MAX_IMAGE_PIXELS = None

def convert_pdf_to_images(pdf_dir, output_dir, dpi=300):
    """
    Convert PDF documents to high-resolution images.

    Args:
        pdf_dir: Directory containing PDF documents
        output_dir: Directory to save images
        dpi: Resolution for image conversion
    """
    os.makedirs(output_dir, exist_ok=True)

    pdf_files = []
    for root, _, files in os.walk(pdf_dir):
        for file in files:
```

```
            if file.lower().endswith('.pdf'):
                pdf_files.append(os.path.join(root, file))

    for pdf_path in tqdm(pdf_files, desc="Converting PDFs to images"):
        pdf_filename = os.path.splitext(os.path.basename(pdf_path))[0]

        doc_dir = os.path.join(output_dir, pdf_filename)
        os.makedirs(doc_dir, exist_ok=True)

        try:
            doc = fitz.open(pdf_path)

            for page_num, page in enumerate(doc):
                pix = page.get_pixmap(matrix=fitz.Matrix(dpi/72, dpi/72))

                image_path = os.path.join(output_dir, f"{pdf_filename}_page_{page_num+1:03d}.png")
                pix.save(image_path)

                print(f"Saved {image_path}")
        except Exception as e:
            print(f"Error processing {pdf_path}: {e}")

    print(f"PDF to image conversion complete. Images saved to {output_dir}")

convert_pdf_to_images(pdf_dir, image_dir)

# Display a sample image
sample_images = []
for root, _, files in os.walk(image_dir):
    for file in files:
        if file.lower().endswith(('.png', '.jpg', '.jpeg')):
            sample_images.append(os.path.join(root, file))
            if len(sample_images) >= 1:
                break
    if len(sample_images) >= 1:
        break

if sample_images:
    plt.figure(figsize=(10, 15))
    img = plt.imread(sample_images[0])
    plt.imshow(img)
    plt.title(f"Sample Document Image: {os.path.basename(sample_images[0])}")
    plt.axis('off')
    plt.show()
```

```
Converting PDFs to images:   0%|          | 0/6 [00:00<?, ?it/s]
```

```
Saved ./output/images\Buendia - Instruccion_page_001.png
Saved ./output/images\Buendia - Instruccion_page_002.png
Saved ./output/images\Buendia - Instruccion_page_003.png
Saved ./output/images\Buendia - Instruccion_page_004.png
Saved ./output/images\Buendia - Instruccion_page_005.png
Saved ./output/images\Buendia - Instruccion_page_006.png
Saved ./output/images\Constituciones sinodales Calahorra 1602_page_001.png
Saved ./output/images\Constituciones sinodales Calahorra 1602_page_002.png
Saved ./output/images\Constituciones sinodales Calahorra 1602_page_003.png
Saved ./output/images\Constituciones sinodales Calahorra 1602_page_004.png
Saved ./output/images\Constituciones sinodales Calahorra 1602_page_005.png
Saved ./output/images\Constituciones sinodales Calahorra 1602_page_006.png
Saved ./output/images\Ezcaray - Vozes_page_001.png
Saved ./output/images\Ezcaray - Vozes_page_002.png
Saved ./output/images\Ezcaray - Vozes_page_003.png
Saved ./output/images\Ezcaray - Vozes_page_004.png
Saved ./output/images\Ezcaray - Vozes_page_005.png
Saved ./output/images\Ezcaray - Vozes_page_006.png
Saved ./output/images\Ezcaray - Vozes_page_007.png
Saved ./output/images\Ezcaray - Vozes_page_008.png
Saved ./output/images\Ezcaray - Vozes_page_009.png
Saved ./output/images\Ezcaray - Vozes_page_010.png
Saved ./output/images\Ezcaray - Vozes_page_011.png
Saved ./output/images\Mendo - Principe perfecto_page_001.png
Saved ./output/images\Mendo - Principe perfecto_page_002.png
Saved ./output/images\Mendo - Principe perfecto_page_003.png
Saved ./output/images\Mendo - Principe perfecto_page_004.png
Saved ./output/images\Mendo - Principe perfecto_page_005.png
Saved ./output/images\Mendo - Principe perfecto_page_006.png
Saved ./output/images\Mendo - Principe perfecto_page_007.png
Saved ./output/images\Mendo - Principe perfecto_page_008.png
Saved ./output/images\Mendo - Principe perfecto_page_009.png
Saved ./output/images\Paredes - Reglas generales_page_001.png
Saved ./output/images\Paredes - Reglas generales_page_002.png
Saved ./output/images\Paredes - Reglas generales_page_003.png
Saved ./output/images\Paredes - Reglas generales_page_004.png
Saved ./output/images\Paredes - Reglas generales_page_005.png
Saved ./output/images\Paredes - Reglas generales_page_006.png
Saved ./output/images\Paredes - Reglas generales_page_007.png
Saved ./output/images\Paredes - Reglas generales_page_008.png
Saved ./output/images\Paredes - Reglas generales_page_009.png
Saved ./output/images\PORCONES.228.35 – 1636_page_001.png
Saved ./output/images\PORCONES.228.35 – 1636_page_002.png
Saved ./output/images\PORCONES.228.35 – 1636_page_003.png
Saved ./output/images\PORCONES.228.35 – 1636_page_004.png
Saved ./output/images\PORCONES.228.35 – 1636_page_005.png
Saved ./output/images\PORCONES.228.35 – 1636_page_006.png
Saved ./output/images\PORCONES.228.35 – 1636_page_007.png
Saved ./output/images\PORCONES.228.35 – 1636_page_008.png
Saved ./output/images\PORCONES.228.35 – 1636_page_009.png
Saved ./output/images\PORCONES.228.35 – 1636_page_010.png
Saved ./output/images\PORCONES.228.35 – 1636_page_011.png
Saved ./output/images\PORCONES.228.35 – 1636_page_012.png
Saved ./output/images\PORCONES.228.35 – 1636_page_013.png
Saved ./output/images\PORCONES.228.35 – 1636_page_014.png
Saved ./output/images\PORCONES.228.35 – 1636_page_015.png
Saved ./output/images\PORCONES.228.35 – 1636_page_016.png
PDF to image conversion complete. Images saved to ./output/images
```

Sample Document Image: Buendia - Instruccion_page_001.png



AL

INFINITAMENTE AMABLE

NIÑO JESUS.

AVos, Dulciſsimo Niño Jesus, que no ſolo os dignaſteis de llamaros Doctor de los Niños, ſino tambien de aſsiſtir como Niño entre los Doctores, ſe conſagra humilde eſta pequeña Inſtruccion de los Niños. Es aſsi, que ella tambien ſe dirige à la juventud; pero à eſta, como recuerdo de lo que aprendió, à los Niños, como preciſa explicacion de lo que deben eſtudiar. Por eſte ſolo titulo es muy vueſtra; y por ſer para Niños, que confiais à la educacion de vueſtra Compañia, lo es mucho mas. En Vos, (Divino Exemplar de todas las virtudes) tienen abreviado el mas ſeguro

Exiſai. 33. 18. Et Luc. 2. 46.

¶ 2

In [ ]:

In [7]:
```python
# 2. Process Transcriptions
def process_transcriptions(transcription_dir, output_dir):
    """
    Process transcription files.

    Args:
        transcription_dir: Directory containing transcription files
        output_dir: Directory to save processed transcriptions
    """
    os.makedirs(output_dir, exist_ok=True)

    transcription_files = []
    for root, _, files in os.walk(transcription_dir):
        for file in files:
            if file.lower().endswith(('.docx', '.doc')):
                transcription_files.append(os.path.join(root, file))

    for doc_path in tqdm(transcription_files, desc="Processing transcriptions"):
        doc_filename = os.path.splitext(os.path.basename(doc_path))[0]

        try:
            doc = docx.Document(doc_path)

            # Extract text
            text = ""
            for para in doc.paragraphs:
                text += para.text + "\n"

            # Normalize text
            text = text.replace("ſ", "s")
            text = text.replace("ç", "z")

            # Save processed
            output_path = os.path.join(output_dir, f"{doc_filename}.json")
            with open(output_path, 'w', encoding='utf-8') as f:
                json.dump({
                    'document': doc_filename,
                    'transcription': text
                }, f, ensure_ascii=False, indent=2)

            print(f"Processed {doc_path} -> {output_path}")
        except Exception as e:
            print(f"Error processing {doc_path}: {e}")

print(f"Transcription processing complete. Results saved to {output_dir}")
```

```
process_transcriptions(transcription_dir, transcriptions_output_dir)

sample_transcriptions = []
for root, _, files in os.walk(transcriptions_output_dir):
    for file in files:
        if file.lower().endswith('.json'):
            sample_transcriptions.append(os.path.join(root, file))
            if len(sample_transcriptions) >= 1:
                break
    if len(sample_transcriptions) >= 1:
        break

if sample_transcriptions:
    with open(sample_transcriptions[0], 'r', encoding='utf-8') as f:
        transcription = json.load(f)

    print(f"Sample Transcription: {os.path.basename(sample_transcriptions[0])}")
    print("First 500 characters:")
    print(transcription['transcription'][:500] + "...")
```

```
Processing transcriptions:   0%|          | 0/6 [00:00<?, ?it/s]
Processed Test transcriptions\Buendia transcription.docx -> ./output/transcriptions\Buendia transcription.json
Processed Test transcriptions\Constituciones sinodales transcription.docx -> ./output/transcriptions\Constituciones sinodales t
ranscription.json
Processed Test transcriptions\Ezcaray transcription.docx -> ./output/transcriptions\Ezcaray transcription.json
Processed Test transcriptions\Mendo transcription.docx -> ./output/transcriptions\Mendo transcription.json
Processed Test transcriptions\Paredes transcription.docx -> ./output/transcriptions\Paredes transcription.json
Processed Test transcriptions\PORCONES.228.35 1636 transcription.docx -> ./output/transcriptions\PORCONES.228.35 1636 transcrip
tion.json
Transcription processing complete. Results saved to ./output/transcriptions
Sample Transcription: Buendia transcription.json
First 500 characters:
NOTES:          u and v are used interchangeably        check against dictionary?
                two types of lowercase “s” -> ‘s’ and ‘s’  both should be transcribed as ‘s’
                accents are inconsistent                should be ignored (except ñ)
                some letters have macrons (¯)           should mean ‘n’ follows, or ‘ue’ after capped q
                some line end hyphens not present       leave words split for now, can decide later
                z old spelling is always modern z       teach AI to always interpret z as z
PDF p1
Al
INFINITAMENTE AMABLE
NIÑO JESUS.
A Vos, Dulcissimo N...
```

In [ ]:

```
# 3. Create Layout Masks
def create_layout_masks(image_dir, output_dir):
    """
    Create layout masks for document images.

    Args:
        image_dir: Directory containing document images
        output_dir: Directory to save layout masks
    """
    os.makedirs(output_dir, exist_ok=True)

    # Get all image files
    image_files = []
    for root, _, files in os.walk(image_dir):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                image_files.append(os.path.join(root, file))

    for image_path in tqdm(image_files, desc="Creating layout masks"):
        try:
            image = Image.open(image_path).convert("RGB")

            cv_image = np.array(image)
            cv_image = cv_image[:, :, ::-1].copy()

            gray = cv2.cvtColor(cv_image, cv2.COLOR_BGR2GRAY)

            blurred = cv2.GaussianBlur(gray, (5, 5), 0)

            _, binary = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

            contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

            mask = np.zeros((image.height, image.width), dtype=np.uint8)

            boxes = []
            for contour in contours:
                x, y, w, h = cv2.boundingRect(contour)

                # Filter small contours (noise)
                if w > 50 and h > 20:
                    mask[y:y+h, x:x+w] = 1
                    boxes.append([x, y, x+w, y+h])
```

```
            # Save mask
            output_path = os.path.join(output_dir, os.path.basename(image_path))
            cv2.imwrite(output_path, mask * 255)  # Scale to 0-255 for visualization

            boxes_json = {
                "image_path": image_path,
                "boxes": boxes
            }
            json_path = output_path.replace('.png', '.json').replace('.jpg', '.json')
            with open(json_path, 'w') as f:
                json.dump(boxes_json, f)

            print(f"Created mask for {image_path} -> {output_path}")
        except Exception as e:
            print(f"Error processing {image_path}: {e}")

    print(f"Layout mask creation complete. Masks saved to {output_dir}")

# Run layout mask creation
create_layout_masks(image_dir, masks_dir)

# Display a sample image and its mask
sample_masks = []
for root, _, files in os.walk(masks_dir):
    for file in files:
        if file.lower().endswith(('.png', '.jpg', '.jpeg')):
            sample_masks.append(os.path.join(root, file))
            if len(sample_masks) >= 1:
                break
    if len(sample_masks) >= 1:
        break

if sample_images and sample_masks:
    plt.figure(figsize=(15, 10))

    plt.subplot(1, 2, 1)
    img = plt.imread(sample_images[0])
    plt.imshow(img)
    plt.title("Original Document")
    plt.axis('off')

    plt.subplot(1, 2, 2)
    mask = plt.imread(sample_masks[0])
    plt.imshow(mask, cmap='gray')
    plt.title("Layout Mask")
    plt.axis('off')

    plt.tight_layout()
    plt.show()
```

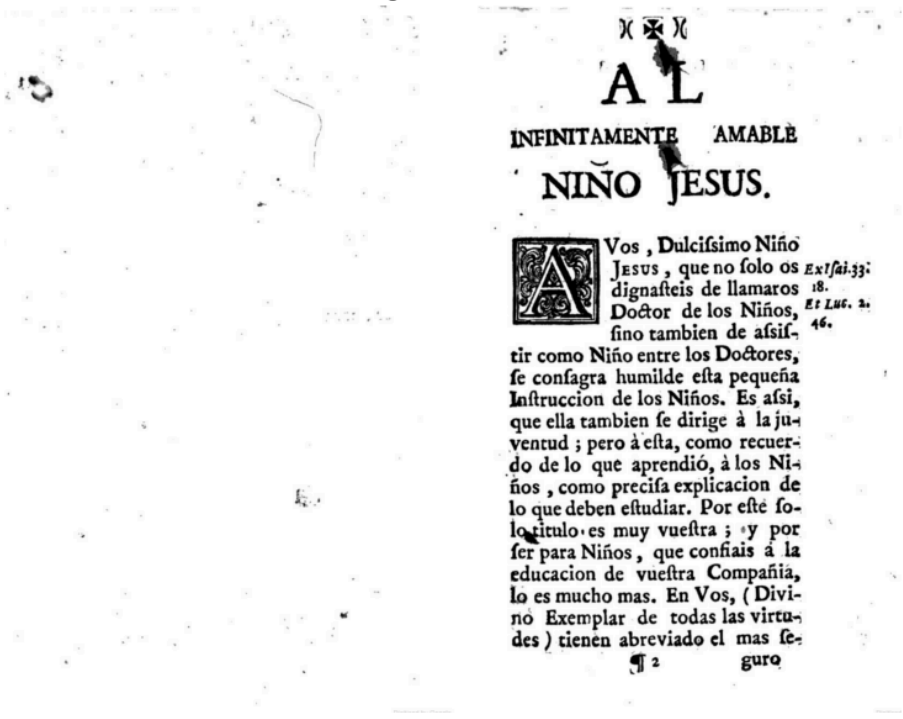
```
Creating layout masks:   0%|          | 0/114 [00:00<?, ?it/s]
```

```
Created mask for ./output/images\Buendia - Instruccion_page_001.png -> ./output/masks\Buendia - Instruccion_page_001.png
Created mask for ./output/images\Buendia - Instruccion_page_002.png -> ./output/masks\Buendia - Instruccion_page_002.png
Created mask for ./output/images\Buendia - Instruccion_page_003.png -> ./output/masks\Buendia - Instruccion_page_003.png
Created mask for ./output/images\Buendia - Instruccion_page_004.png -> ./output/masks\Buendia - Instruccion_page_004.png
Created mask for ./output/images\Buendia - Instruccion_page_005.png -> ./output/masks\Buendia - Instruccion_page_005.png
Created mask for ./output/images\Buendia - Instruccion_page_006.png -> ./output/masks\Buendia - Instruccion_page_006.png
Created mask for ./output/images\Constituciones sinodales Calahorra 1602_page_001.png -> ./output/masks\Constituciones sinodale
s Calahorra 1602_page_001.png
Created mask for ./output/images\Constituciones sinodales Calahorra 1602_page_002.png -> ./output/masks\Constituciones sinodale
s Calahorra 1602_page_002.png
Created mask for ./output/images\Constituciones sinodales Calahorra 1602_page_003.png -> ./output/masks\Constituciones sinodale
s Calahorra 1602_page_003.png
Created mask for ./output/images\Constituciones sinodales Calahorra 1602_page_004.png -> ./output/masks\Constituciones sinodale
s Calahorra 1602_page_004.png
Created mask for ./output/images\Constituciones sinodales Calahorra 1602_page_005.png -> ./output/masks\Constituciones sinodale
s Calahorra 1602_page_005.png
Created mask for ./output/images\Constituciones sinodales Calahorra 1602_page_006.png -> ./output/masks\Constituciones sinodale
s Calahorra 1602_page_006.png
Created mask for ./output/images\Ezcaray - Vozes_page_001.png -> ./output/masks\Ezcaray - Vozes_page_001.png
Created mask for ./output/images\Ezcaray - Vozes_page_002.png -> ./output/masks\Ezcaray - Vozes_page_002.png
Created mask for ./output/images\Ezcaray - Vozes_page_003.png -> ./output/masks\Ezcaray - Vozes_page_003.png
Created mask for ./output/images\Ezcaray - Vozes_page_004.png -> ./output/masks\Ezcaray - Vozes_page_004.png
Created mask for ./output/images\Ezcaray - Vozes_page_005.png -> ./output/masks\Ezcaray - Vozes_page_005.png
Created mask for ./output/images\Ezcaray - Vozes_page_006.png -> ./output/masks\Ezcaray - Vozes_page_006.png
Created mask for ./output/images\Ezcaray - Vozes_page_007.png -> ./output/masks\Ezcaray - Vozes_page_007.png
Created mask for ./output/images\Ezcaray - Vozes_page_008.png -> ./output/masks\Ezcaray - Vozes_page_008.png
Created mask for ./output/images\Ezcaray - Vozes_page_009.png -> ./output/masks\Ezcaray - Vozes_page_009.png
Created mask for ./output/images\Ezcaray - Vozes_page_010.png -> ./output/masks\Ezcaray - Vozes_page_010.png
Created mask for ./output/images\Ezcaray - Vozes_page_011.png -> ./output/masks\Ezcaray - Vozes_page_011.png
Created mask for ./output/images\Mendo - Principe perfecto_page_001.png -> ./output/masks\Mendo - Principe perfecto_page_001.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_002.png -> ./output/masks\Mendo - Principe perfecto_page_002.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_003.png -> ./output/masks\Mendo - Principe perfecto_page_003.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_004.png -> ./output/masks\Mendo - Principe perfecto_page_004.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_005.png -> ./output/masks\Mendo - Principe perfecto_page_005.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_006.png -> ./output/masks\Mendo - Principe perfecto_page_006.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_007.png -> ./output/masks\Mendo - Principe perfecto_page_007.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_008.png -> ./output/masks\Mendo - Principe perfecto_page_008.pn
g
Created mask for ./output/images\Mendo - Principe perfecto_page_009.png -> ./output/masks\Mendo - Principe perfecto_page_009.pn
g
Created mask for ./output/images\Paredes - Reglas generales_page_001.png -> ./output/masks\Paredes - Reglas generales_page_001.
png
Created mask for ./output/images\Paredes - Reglas generales_page_002.png -> ./output/masks\Paredes - Reglas generales_page_002.
png
Created mask for ./output/images\Paredes - Reglas generales_page_003.png -> ./output/masks\Paredes - Reglas generales_page_003.
png
Created mask for ./output/images\Paredes - Reglas generales_page_004.png -> ./output/masks\Paredes - Reglas generales_page_004.
png
Created mask for ./output/images\Paredes - Reglas generales_page_005.png -> ./output/masks\Paredes - Reglas generales_page_005.
png
Created mask for ./output/images\Paredes - Reglas generales_page_006.png -> ./output/masks\Paredes - Reglas generales_page_006.
png
Created mask for ./output/images\Paredes - Reglas generales_page_007.png -> ./output/masks\Paredes - Reglas generales_page_007.
png
Created mask for ./output/images\Paredes - Reglas generales_page_008.png -> ./output/masks\Paredes - Reglas generales_page_008.
png
Created mask for ./output/images\Paredes - Reglas generales_page_009.png -> ./output/masks\Paredes - Reglas generales_page_009.
png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_001.png -> ./output/masks\PORCONES.228.35 – 1636_page_001.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_002.png -> ./output/masks\PORCONES.228.35 – 1636_page_002.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_003.png -> ./output/masks\PORCONES.228.35 – 1636_page_003.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_004.png -> ./output/masks\PORCONES.228.35 – 1636_page_004.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_005.png -> ./output/masks\PORCONES.228.35 – 1636_page_005.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_006.png -> ./output/masks\PORCONES.228.35 – 1636_page_006.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_007.png -> ./output/masks\PORCONES.228.35 – 1636_page_007.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_008.png -> ./output/masks\PORCONES.228.35 – 1636_page_008.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_009.png -> ./output/masks\PORCONES.228.35 – 1636_page_009.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_010.png -> ./output/masks\PORCONES.228.35 – 1636_page_010.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_011.png -> ./output/masks\PORCONES.228.35 – 1636_page_011.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_012.png -> ./output/masks\PORCONES.228.35 – 1636_page_012.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_013.png -> ./output/masks\PORCONES.228.35 – 1636_page_013.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_014.png -> ./output/masks\PORCONES.228.35 – 1636_page_014.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_015.png -> ./output/masks\PORCONES.228.35 – 1636_page_015.png
Created mask for ./output/images\PORCONES.228.35 – 1636_page_016.png -> ./output/masks\PORCONES.228.35 – 1636_page_016.png
Created mask for ./output/images\Buendia\Buendia - Instruccion_page_001.png -> ./output/masks\Buendia - Instruccion_page_001.pn
g
Created mask for ./output/images\Buendia\Buendia - Instruccion_page_002.png -> ./output/masks\Buendia - Instruccion_page_002.pn
g
Created mask for ./output/images\Buendia\Buendia - Instruccion_page_003.png -> ./output/masks\Buendia - Instruccion_page_003.pn
g
```

```
Created mask for ./output/images\Buendia\Buendia - Instruccion_page_004.png -> ./output/masks\Buendia - Instruccion_page_004.pn
g
Created mask for ./output/images\Buendia\Buendia - Instruccion_page_005.png -> ./output/masks\Buendia - Instruccion_page_005.pn
g
Created mask for ./output/images\Buendia\Buendia - Instruccion_page_006.png -> ./output/masks\Buendia - Instruccion_page_006.pn
g
Created mask for ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_001.png -> ./output/masks\Constitu
ciones sinodales Calahorra 1602_page_001.png
Created mask for ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_002.png -> ./output/masks\Constitu
ciones sinodales Calahorra 1602_page_002.png
Created mask for ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_003.png -> ./output/masks\Constitu
ciones sinodales Calahorra 1602_page_003.png
Created mask for ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_004.png -> ./output/masks\Constitu
ciones sinodales Calahorra 1602_page_004.png
Created mask for ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_005.png -> ./output/masks\Constitu
ciones sinodales Calahorra 1602_page_005.png
Created mask for ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_006.png -> ./output/masks\Constitu
ciones sinodales Calahorra 1602_page_006.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_001.png -> ./output/masks\Ezcaray - Vozes_page_001.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_002.png -> ./output/masks\Ezcaray - Vozes_page_002.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_003.png -> ./output/masks\Ezcaray - Vozes_page_003.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_004.png -> ./output/masks\Ezcaray - Vozes_page_004.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_005.png -> ./output/masks\Ezcaray - Vozes_page_005.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_006.png -> ./output/masks\Ezcaray - Vozes_page_006.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_007.png -> ./output/masks\Ezcaray - Vozes_page_007.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_008.png -> ./output/masks\Ezcaray - Vozes_page_008.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_009.png -> ./output/masks\Ezcaray - Vozes_page_009.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_010.png -> ./output/masks\Ezcaray - Vozes_page_010.png
Created mask for ./output/images\Ezcaray\Ezcaray - Vozes_page_011.png -> ./output/masks\Ezcaray - Vozes_page_011.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_001.png -> ./output/masks\Mendo - Principe perfecto_page_
001.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_002.png -> ./output/masks\Mendo - Principe perfecto_page_
002.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_003.png -> ./output/masks\Mendo - Principe perfecto_page_
003.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_004.png -> ./output/masks\Mendo - Principe perfecto_page_
004.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_005.png -> ./output/masks\Mendo - Principe perfecto_page_
005.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_006.png -> ./output/masks\Mendo - Principe perfecto_page_
006.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_007.png -> ./output/masks\Mendo - Principe perfecto_page_
007.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_008.png -> ./output/masks\Mendo - Principe perfecto_page_
008.png
Created mask for ./output/images\Mendo\Mendo - Principe perfecto_page_009.png -> ./output/masks\Mendo - Principe perfecto_page_
009.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_001.png -> ./output/masks\Paredes - Reglas generales_p
age_001.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_002.png -> ./output/masks\Paredes - Reglas generales_p
age_002.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_003.png -> ./output/masks\Paredes - Reglas generales_p
age_003.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_004.png -> ./output/masks\Paredes - Reglas generales_p
age_004.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_005.png -> ./output/masks\Paredes - Reglas generales_p
age_005.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_006.png -> ./output/masks\Paredes - Reglas generales_p
age_006.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_007.png -> ./output/masks\Paredes - Reglas generales_p
age_007.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_008.png -> ./output/masks\Paredes - Reglas generales_p
age_008.png
Created mask for ./output/images\Paredes\Paredes - Reglas generales_page_009.png -> ./output/masks\Paredes - Reglas generales_p
age_009.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_001.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_001.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_002.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_002.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_003.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_003.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_004.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_004.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_005.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_005.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_006.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_006.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_007.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_007.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_008.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_008.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_009.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_009.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_010.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_010.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_011.png -> ./output/masks\PORCONES.228.35 – 1636_p
age_011.png
```

```
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_012.png -> ./output/masks\PORCONES.228.35 – 1636_page_012.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_013.png -> ./output/masks\PORCONES.228.35 – 1636_page_013.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_014.png -> ./output/masks\PORCONES.228.35 – 1636_page_014.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_015.png -> ./output/masks\PORCONES.228.35 – 1636_page_015.png
Created mask for ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_016.png -> ./output/masks\PORCONES.228.35 – 1636_page_016.png
Layout mask creation complete. Masks saved to ./output/masks
```

Original Document



AL

INFINITAMENTE AMABLE

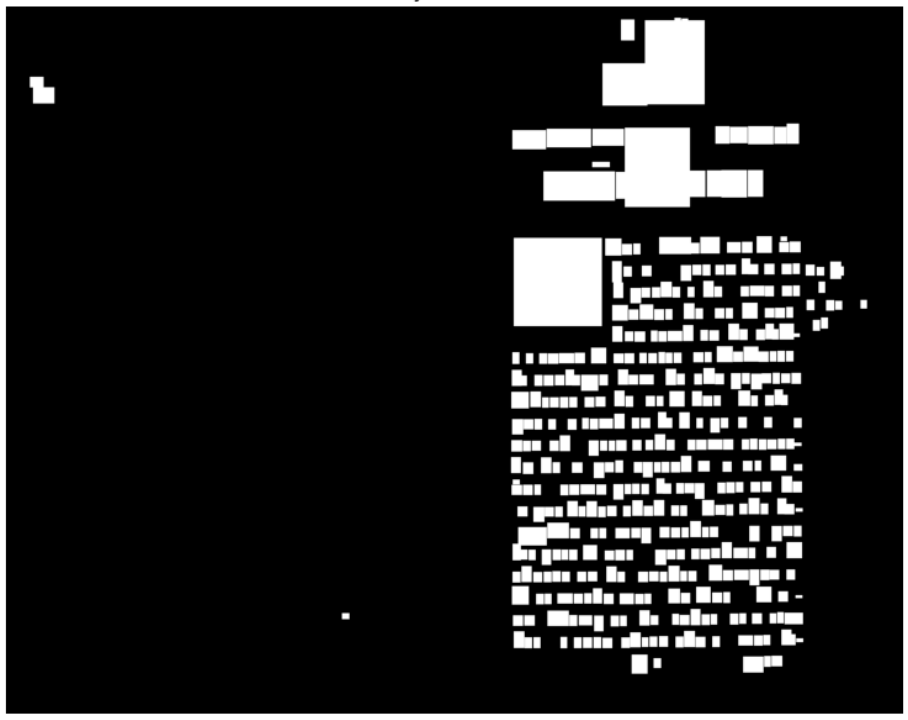
NIÑO JESUS.

AVos, Dulciſsimo Niño Jesus, que no ſolo os dignaſteis de llamaros Doctor de los Niños, ſino tambien de aſsiſtir como Niño entre los Doctores, ſe conſagra humilde eſta pequeña Inſtruccion de los Niños. Es aſsi, que ella tambien ſe dirige à la juventud; pero à eſta, como recuerdo de lo que aprendió, à los Niños, como preciſa explicacion de lo que deben eſtudiar. Por eſte ſolo titulo es muy vueſtra; y por ſer para Niños, que confiais à la educacion de vueſtra Compañia, lo es mucho mas. En Vos, (Divino Exemplar de todas las virtudes) tienen abreviado el mas ſeguro

Exſai. 33. 18. Et Luc. 2. 46.

¶ 2

Layout Mask



In [9]:

```python
# 4. Layout Model Definition
class LayoutSegmentationModel(nn.Module):
    def __init__(self, model_type="unet", encoder_name="resnet34", num_classes=1):
        """
        Layout segmentation model.

        Args:
            model_type: Type of segmentation model
            encoder_name: Name of the encoder backbone
            num_classes: Number of output classes
        """
        super(LayoutSegmentationModel, self).__init__()

        if model_type == "unet":
            self.model = smp.Unet(
                encoder_name=encoder_name,
                encoder_weights="imagenet",
                in_channels=3,
                classes=num_classes,
                activation="sigmoid"
            )
        elif model_type == "fpn":
            self.model = smp.FPN(
                encoder_name=encoder_name,
                encoder_weights="imagenet",
                in_channels=3,
                classes=num_classes,
                activation="sigmoid"
            )
        else:
            raise ValueError(f"Unsupported model type: {model_type}")

    def forward(self, x):
        """Forward pass"""
        return self.model(x)

# Loss function
def bce_dice_loss(y_pred, y_true, smooth=1e-6):
    """
    Combined binary cross-entropy and Dice loss.

    Args:
        y_pred: Predicted mask
        y_true: Ground truth mask
        smooth: Smoothing factor

    Returns:
        Combined loss
    """
    bce = nn.BCELoss()(y_pred, y_true)

    y_pred_flat = y_pred.view(-1)
    y_true_flat = y_true.view(-1)
```

```
    intersection = (y_pred_flat * y_true_flat).sum()
    dice = 1 - (2. * intersection + smooth) / (y_pred_flat.sum() + y_true_flat.sum() + smooth)

    return bce + dice

# Dataset class
class LayoutDataset(Dataset):
    def __init__(self, image_dir, mask_dir, transform=None):
        """
        Dataset for layout segmentation.

        Args:
            image_dir: Directory containing document images
            mask_dir: Directory containing layout masks
            transform: Albumentations transformations
        """
        self.image_dir = image_dir
        self.mask_dir = mask_dir
        self.transform = transform

        # Get all image files
        self.image_files = []
        for root, _, files in os.walk(image_dir):
            for file in files:
                if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                    self.image_files.append(os.path.join(root, file))

        # Get all mask files
        self.mask_files = []
        for image_path in self.image_files:
            # Extract filename without extension
            filename = os.path.basename(image_path)

            # Construct mask path
            mask_path = os.path.join(mask_dir, filename)

            # Check if mask exists
            if os.path.exists(mask_path):
                self.mask_files.append(mask_path)
            else:
                print(f"Warning: Mask not found for {image_path}")
                # Use a placeholder mask path
                self.mask_files.append(None)

        # Make sure we have the same number of images and masks
        assert len(self.image_files) == len(self.mask_files), "Number of images and masks must be the same"

    def __len__(self):
        return len(self.image_files)

    def __getitem__(self, idx):
        """Get a sample from the dataset"""
        # Get image and mask paths
        image_path = self.image_files[idx]
        mask_path = self.mask_files[idx]

        # Skip if mask is None
        if mask_path is None:
            # Return a dummy sample
            dummy_image = np.zeros((384, 384, 3), dtype=np.uint8)
            dummy_mask = np.zeros((384, 384), dtype=np.float32)

            if self.transform:
                augmented = self.transform(image=dummy_image, mask=dummy_mask)
                return augmented['image'], augmented['mask'].float().unsqueeze(0)
            return torch.from_numpy(dummy_image.transpose(2, 0, 1)), torch.from_numpy(dummy_mask[None, :, :]).float()

        # Load image and mask
        try:
            image = cv2.imread(image_path)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

            mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
            mask = mask / 255.0  # Normalize to 0-1
            mask = mask.astype(np.float32)  # Explicitly convert to float32
        except Exception as e:
            print(f"Error loading {image_path} or {mask_path}: {e}")
            # Return a dummy sample
            dummy_image = np.zeros((384, 384, 3), dtype=np.uint8)
            dummy_mask = np.zeros((384, 384), dtype=np.float32)

            if self.transform:
                augmented = self.transform(image=dummy_image, mask=dummy_mask)
                return augmented['image'], augmented['mask'].float().unsqueeze(0)
            return torch.from_numpy(dummy_image.transpose(2, 0, 1)), torch.from_numpy(dummy_mask[None, :, :]).float()
```

```
        # Apply transformations
        if self.transform:
            augmented = self.transform(image=image, mask=mask)
            image = augmented['image']
            mask = augmented['mask'].float().unsqueeze(0)
        else:
            # Convert to torch tensors
            image = torch.from_numpy(image.transpose(2, 0, 1))
            mask = torch.from_numpy(mask[None, :, :]).float()

        return image, mask

# Transformations
def get_train_transform():
    """Get training transformations with augmentations"""
    return A.Compose([
        A.Resize(512, 512),
        A.HorizontalFlip(p=0.2),
        A.VerticalFlip(p=0.2),
        A.RandomRotate90(p=0.2),
        A.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.1, rotate_limit=15, p=0.3),
        A.OneOf([
            A.GaussNoise(p=0.5),
            A.GaussianBlur(p=0.5),
        ], p=0.3),
        A.OneOf([
            A.RandomBrightnessContrast(p=0.5),
            A.RandomGamma(p=0.5),
        ], p=0.3),
        A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
        ToTensorV2(),
    ])

def get_val_transform():
    """Get validation transformations"""
    return A.Compose([
        A.Resize(512, 512),
        A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
        ToTensorV2(),
    ])

# Dataloaders
def get_dataloaders(image_dir, mask_dir, batch_size=8, val_split=0.2):
    """
    Get training and validation dataloaders.

    Args:
        image_dir: Directory containing document images
        mask_dir: Directory containing layout masks
        batch_size: Batch size for training
        val_split: Validation split ratio

    Returns:
        train_loader, val_loader
    """
    # Create dataset
    dataset = LayoutDataset(
        image_dir=image_dir,
        mask_dir=mask_dir,
        transform=get_train_transform()
    )

    # Split dataset
    val_size = int(len(dataset) * val_split)
    train_size = len(dataset) - val_size

    train_dataset, val_dataset = torch.utils.data.random_split(dataset, [train_size, val_size])

    # Update transforms for validation dataset
    val_dataset.dataset.transform = get_val_transform()

    # Create dataloaders
    train_loader = DataLoader(
        train_dataset,
        batch_size=batch_size,
        shuffle=True,
        num_workers=4,
        pin_memory=True
    )

    val_loader = DataLoader(
        val_dataset,
        batch_size=batch_size,
        shuffle=False,
        num_workers=4,
        pin_memory=True
    )
```

```
    return train_loader, val_loader
```

In [12]:
```
# 5. Layout Model Training
def train_layout_model(image_dir, mask_dir, output_dir, model_type="unet", encoder_name="resnet34",
                       batch_size=8, num_epochs=50, learning_rate=3e-4):
    """
    Train layout segmentation model.
    """
    # Create output directory
    os.makedirs(output_dir, exist_ok=True)

    # Set device
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print(f"Using device: {device}")

    # Get dataloaders with reduced num_workers
    train_loader, val_loader = get_dataloaders(
        image_dir=image_dir,
        mask_dir=mask_dir,
        batch_size=batch_size,
        num_workers=0  # Set to 0 to avoid multiprocessing issues
    )

    print(f"Training samples: {len(train_loader.dataset)}")
    print(f"Validation samples: {len(val_loader.dataset)}")

    # Create model
    model = LayoutSegmentationModel(
        model_type=model_type,
        encoder_name=encoder_name
    ).to(device)

    # Optimizer
    optimizer = optim.Adam(model.parameters(), lr=learning_rate)

    # Learning rate scheduler
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(
        optimizer, mode='min', factor=0.5, patience=5, verbose=True
    )

    # Training loop
    best_val_loss = float('inf')
    train_losses = []
    val_losses = []
    patience = 10
    patience_counter = 0

    for epoch in range(num_epochs):
        # Training phase
        model.train()
        epoch_loss = 0

        for images, masks in tqdm(train_loader, desc=f"Epoch {epoch+1}/{num_epochs} - Training"):
            images = images.to(device)
            masks = masks.to(device)

            # Forward pass
            outputs = model(images)
            loss = bce_dice_loss(outputs, masks)

            # Backward pass
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            epoch_loss += loss.item()

        avg_train_loss = epoch_loss / len(train_loader)
        train_losses.append(avg_train_loss)

        # Validation phase
        model.eval()
        val_loss = 0
        val_iou = 0

        with torch.no_grad():
            for images, masks in tqdm(val_loader, desc=f"Epoch {epoch+1}/{num_epochs} - Validation"):
                images = images.to(device)
                masks = masks.to(device)

                # Forward pass
                outputs = model(images)
                loss = bce_dice_loss(outputs, masks)
                val_loss += loss.item()

                # Calculate IoU
```

```
            pred = (outputs > 0.5).float()
            intersection = (pred * masks).sum((1, 2, 3))
            union = pred.sum((1, 2, 3)) + masks.sum((1, 2, 3)) - intersection
            batch_iou = (intersection + 1e-6) / (union + 1e-6)
            val_iou += batch_iou.mean().item()

        avg_val_loss = val_loss / len(val_loader)
        avg_val_iou = val_iou / len(val_loader)
        val_losses.append(avg_val_loss)

        # Update learning rate
        scheduler.step(avg_val_loss)

        print(f"Epoch {epoch+1}/{num_epochs}")
        print(f"Train Loss: {avg_train_loss:.4f}")
        print(f"Val Loss: {avg_val_loss:.4f}")
        print(f"Val IoU: {avg_val_iou:.4f}")

        # Save best model
        if avg_val_loss < best_val_loss:
            best_val_loss = avg_val_loss
            torch.save(model.state_dict(), os.path.join(output_dir, "best_model.pth"))
            print(f"Saved best model with validation loss: {best_val_loss:.4f}")
            patience_counter = 0
        else:
            patience_counter += 1
            if patience_counter >= patience:
                print(f"Early stopping triggered after {epoch + 1} epochs")
                break

    # Plot training history
    plt.figure(figsize=(10, 5))
    plt.plot(train_losses, label='Training Loss')
    plt.plot(val_losses, label='Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training History')
    plt.legend()
    plt.savefig(os.path.join(output_dir, 'training_history.png'))
    plt.close()

    return model

# Update the get_dataloaders function
def get_dataloaders(image_dir, mask_dir, batch_size=8, val_split=0.2, num_workers=0):
    """
    Get training and validation dataloaders.

    Args:
        image_dir: Directory containing document images
        mask_dir: Directory containing layout masks
        batch_size: Batch size for training
        val_split: Validation split ratio
        num_workers: Number of worker processes (set to 0 for troubleshooting)

    Returns:
        train_loader, val_loader
    """
    # Create dataset
    dataset = LayoutDataset(
        image_dir=image_dir,
        mask_dir=mask_dir,
        transform=get_train_transform()
    )

    # Split dataset
    val_size = int(len(dataset) * val_split)
    train_size = len(dataset) - val_size

    train_dataset, val_dataset = torch.utils.data.random_split(dataset, [train_size, val_size])

    # Update transforms for validation dataset
    val_dataset.dataset.transform = get_val_transform()

    # Create dataloaders with num_workers=0
    train_loader = DataLoader(
        train_dataset,
        batch_size=batch_size,
        shuffle=True,
        num_workers=num_workers,
        pin_memory=True
    )

    val_loader = DataLoader(
        val_dataset,
        batch_size=batch_size,
        shuffle=False,
```

```
        num_workers=num_workers,
        pin_memory=True
    )

    return train_loader, val_loader

# Train layout model
layout_model = train_layout_model(
    image_dir=image_dir,
    mask_dir=masks_dir,
    output_dir=layout_model_dir,
    model_type="unet",
    encoder_name="resnet34",
    batch_size=4,  # Adjust based on your GPU memory
    num_epochs=20,
    learning_rate=3e-4
)
```

```
Using device: cuda
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_001.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_002.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_003.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_004.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_005.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_006.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_007.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_008.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_009.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_010.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_011.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_012.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_013.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_014.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_015.png
Warning: Mask not found for ./output/images\PORCONES.228.35 - 1636_page_016.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_001.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_002.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_003.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_004.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_005.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_006.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_007.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_008.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_009.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_010.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_011.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_012.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_013.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_014.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_015.png
Warning: Mask not found for ./output/images\PORCONES.228.35\PORCONES.228.35 - 1636_page_016.png
Training samples: 92
Validation samples: 22
Epoch 1/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 1/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 1/20
Train Loss: 1.1264
Val Loss: 1.4689
Val IoU: 0.1710
Saved best model with validation loss: 1.4689
Epoch 2/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 2/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 2/20
Train Loss: 0.8851
Val Loss: 0.9159
Val IoU: 0.2389
Saved best model with validation loss: 0.9159
Epoch 3/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 3/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 3/20
Train Loss: 0.7913
Val Loss: 0.8650
Val IoU: 0.2667
Saved best model with validation loss: 0.8650
Epoch 4/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 4/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 4/20
Train Loss: 0.6992
Val Loss: 0.7654
Val IoU: 0.7259
Saved best model with validation loss: 0.7654
Epoch 5/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 5/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 5/20
Train Loss: 0.6388
Val Loss: 0.7056
Val IoU: 0.7397
Saved best model with validation loss: 0.7056
```

```
Epoch 6/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 6/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 6/20
Train Loss: 0.5976
Val Loss: 0.7754
Val IoU: 0.2988
Epoch 7/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 7/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 7/20
Train Loss: 0.5563
Val Loss: 0.6328
Val IoU: 0.7546
Saved best model with validation loss: 0.6328
Epoch 8/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 8/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 8/20
Train Loss: 0.4921
Val Loss: 0.6087
Val IoU: 0.7670
Saved best model with validation loss: 0.6087
Epoch 9/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 9/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 9/20
Train Loss: 0.4587
Val Loss: 0.5683
Val IoU: 0.7839
Saved best model with validation loss: 0.5683
Epoch 10/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 10/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 10/20
Train Loss: 0.4498
Val Loss: 0.5452
Val IoU: 0.7908
Saved best model with validation loss: 0.5452
Epoch 11/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 11/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 11/20
Train Loss: 0.4448
Val Loss: 0.6078
Val IoU: 0.7574
Epoch 12/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 12/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 12/20
Train Loss: 0.4450
Val Loss: 0.5456
Val IoU: 0.7862
Epoch 13/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 13/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 13/20
Train Loss: 0.4027
Val Loss: 0.5085
Val IoU: 0.8062
Saved best model with validation loss: 0.5085
Epoch 14/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 14/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 14/20
Train Loss: 0.3707
Val Loss: 0.5107
Val IoU: 0.8044
Epoch 15/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 15/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 15/20
Train Loss: 0.3770
Val Loss: 0.4888
Val IoU: 0.8222
Saved best model with validation loss: 0.4888
Epoch 16/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 16/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 16/20
Train Loss: 0.3496
Val Loss: 0.6010
Val IoU: 0.7540
Epoch 17/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 17/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 17/20
Train Loss: 0.3282
Val Loss: 0.4864
Val IoU: 0.8221
Saved best model with validation loss: 0.4864
Epoch 18/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 18/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 18/20
Train Loss: 0.3176
Val Loss: 0.4768
Val IoU: 0.8290
Saved best model with validation loss: 0.4768
Epoch 19/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
```

```
Epoch 19/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 19/20
Train Loss: 0.2838
Val Loss: 0.4629
Val IoU: 0.8360
Saved best model with validation loss: 0.4629
Epoch 20/20 - Training:   0%|          | 0/23 [00:00<?, ?it/s]
Epoch 20/20 - Validation:   0%|          | 0/6 [00:00<?, ?it/s]
Epoch 20/20
Train Loss: 0.3284
Val Loss: 0.4944
Val IoU: 0.8135
```

In [13]:

```python
# 6. Layout Prediction and Region Extraction
def predict_layout(model_path, image_path, device):
    """
    Predict layout mask for a document image.

    Args:
        model_path: Path to the trained model
        image_path: Path to document image
        device: Device to run inference on

    Returns:
        Predicted mask
    """
    # Load model
    model = LayoutSegmentationModel()
    model.load_state_dict(torch.load(model_path, map_location=device))
    model.to(device)
    model.eval()

    # Load image
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Apply transformations
    transform = get_val_transform()
    augmented = transform(image=image)
    image_tensor = augmented['image'].unsqueeze(0).to(device)

    # Predict mask
    with torch.no_grad():
        output = model(image_tensor)
        mask = output.squeeze().cpu().numpy()

    # Resize to original size
    mask = cv2.resize(mask, (image.shape[1], image.shape[0]))

    # Convert to binary mask
    mask = (mask > 0.5).astype(np.uint8) * 255

    return mask

def extract_regions(image_path, mask, output_dir):
    """
    Extract text regions from document image using mask.

    Args:
        image_path: Path to document image
        mask: Binary mask
        output_dir: Directory to save extracted regions

    Returns:
        List of region paths and coordinates
    """
    # Create output directory
    os.makedirs(output_dir, exist_ok=True)

    # Load image
    image = cv2.imread(image_path)

    # Find contours in mask
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Extract regions
    regions = []
    for i, contour in enumerate(contours):
        # Get bounding box
        x, y, w, h = cv2.boundingRect(contour)

        # Filter small regions
        if w < 20 or h < 20:
            continue

        # Extract region
        region = image[y:y+h, x:x+w]
```

```
        # Save region
        region_path = os.path.join(output_dir, f"region_{i:03d}.png")
        cv2.imwrite(region_path, region)

        # Add to regions list
        regions.append({
            'path': region_path,
            'bbox': [x, y, w, h]
        })

    return regions

def process_document_images(model_path, image_dir, output_dir):
    """
    Process document images with layout model.

    Args:
        model_path: Path to the trained model
        image_dir: Directory containing document images
        output_dir: Directory to save results
    """
    # Create output directories
    os.makedirs(output_dir, exist_ok=True)
    os.makedirs(os.path.join(output_dir, "masks"), exist_ok=True)
    os.makedirs(os.path.join(output_dir, "regions"), exist_ok=True)

    # Set device
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    # Get all image files
    image_files = []
    for root, _, files in os.walk(image_dir):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                image_files.append(os.path.join(root, file))

    # Process each image
    for image_path in tqdm(image_files, desc="Processing images"):
        try:
            # Get image filename
            filename = os.path.basename(image_path)
            name_without_ext = os.path.splitext(filename)[0]

            # Predict layout mask
            mask = predict_layout(model_path, image_path, device)

            # Save mask
            mask_path = os.path.join(output_dir, "masks", filename)
            cv2.imwrite(mask_path, mask)

            # Extract regions
            regions_dir = os.path.join(output_dir, "regions", name_without_ext)
            regions = extract_regions(image_path, mask, regions_dir)

            # Save regions metadata
            metadata = {
                'image_path': image_path,
                'mask_path': mask_path,
                'regions': regions
            }

            metadata_path = os.path.join(output_dir, f"{name_without_ext}_regions.json")
            with open(metadata_path, 'w') as f:
                json.dump(metadata, f, indent=2)

            print(f"Processed {image_path} -> {len(regions)} regions")
        except Exception as e:
            print(f"Error processing {image_path}: {e}")

    print(f"Document processing complete. Results saved to {output_dir}")

# Process document images with the trained model
process_document_images(
    model_path=os.path.join(layout_model_dir, "best_model.pth"),
    image_dir=image_dir,
    output_dir=regions_dir
)
```

```
Processing images:   0%|          | 0/114 [00:00<?, ?it/s]
```

```
Processed ./output/images\Buendia - Instruccion_page_001.png -> 128 regions
Processed ./output/images\Buendia - Instruccion_page_002.png -> 401 regions
Processed ./output/images\Buendia - Instruccion_page_003.png -> 201 regions
Processed ./output/images\Buendia - Instruccion_page_004.png -> 255 regions
Processed ./output/images\Buendia - Instruccion_page_005.png -> 351 regions
Processed ./output/images\Buendia - Instruccion_page_006.png -> 369 regions
Processed ./output/images\Constituciones sinodales Calahorra 1602_page_001.png -> 46 regions
Processed ./output/images\Constituciones sinodales Calahorra 1602_page_002.png -> 76 regions
Processed ./output/images\Constituciones sinodales Calahorra 1602_page_003.png -> 87 regions
Processed ./output/images\Constituciones sinodales Calahorra 1602_page_004.png -> 155 regions
Processed ./output/images\Constituciones sinodales Calahorra 1602_page_005.png -> 43 regions
Processed ./output/images\Constituciones sinodales Calahorra 1602_page_006.png -> 54 regions
Processed ./output/images\Ezcaray - Vozes_page_001.png -> 169 regions
Processed ./output/images\Ezcaray - Vozes_page_002.png -> 125 regions
Processed ./output/images\Ezcaray - Vozes_page_003.png -> 205 regions
Processed ./output/images\Ezcaray - Vozes_page_004.png -> 188 regions
Processed ./output/images\Ezcaray - Vozes_page_005.png -> 223 regions
Processed ./output/images\Ezcaray - Vozes_page_006.png -> 230 regions
Processed ./output/images\Ezcaray - Vozes_page_007.png -> 206 regions
Processed ./output/images\Ezcaray - Vozes_page_008.png -> 215 regions
Processed ./output/images\Ezcaray - Vozes_page_009.png -> 223 regions
Processed ./output/images\Ezcaray - Vozes_page_010.png -> 216 regions
Processed ./output/images\Ezcaray - Vozes_page_011.png -> 100 regions
Processed ./output/images\Mendo - Principe perfecto_page_001.png -> 143 regions
Processed ./output/images\Mendo - Principe perfecto_page_002.png -> 89 regions
Processed ./output/images\Mendo - Principe perfecto_page_003.png -> 95 regions
Processed ./output/images\Mendo - Principe perfecto_page_004.png -> 108 regions
Processed ./output/images\Mendo - Principe perfecto_page_005.png -> 91 regions
Processed ./output/images\Mendo - Principe perfecto_page_006.png -> 186 regions
Processed ./output/images\Mendo - Principe perfecto_page_007.png -> 87 regions
Processed ./output/images\Mendo - Principe perfecto_page_008.png -> 185 regions
Processed ./output/images\Mendo - Principe perfecto_page_009.png -> 70 regions
Processed ./output/images\Paredes - Reglas generales_page_001.png -> 542 regions
Processed ./output/images\Paredes - Reglas generales_page_002.png -> 615 regions
Processed ./output/images\Paredes - Reglas generales_page_003.png -> 673 regions
Processed ./output/images\Paredes - Reglas generales_page_004.png -> 757 regions
Processed ./output/images\Paredes - Reglas generales_page_005.png -> 689 regions
Processed ./output/images\Paredes - Reglas generales_page_006.png -> 542 regions
Processed ./output/images\Paredes - Reglas generales_page_007.png -> 696 regions
Processed ./output/images\Paredes - Reglas generales_page_008.png -> 689 regions
Processed ./output/images\Paredes - Reglas generales_page_009.png -> 775 regions
Error processing ./output/images\PORCONES.228.35 – 1636_page_001.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_002.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_003.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_004.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_005.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_006.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_007.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_008.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_009.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_010.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_011.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_012.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_013.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_014.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_015.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35 – 1636_page_016.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\mo
```

```
dules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Processed ./output/images\Buendia\Buendia - Instruccion_page_001.png -> 128 regions
Processed ./output/images\Buendia\Buendia - Instruccion_page_002.png -> 401 regions
Processed ./output/images\Buendia\Buendia - Instruccion_page_003.png -> 201 regions
Processed ./output/images\Buendia\Buendia - Instruccion_page_004.png -> 255 regions
Processed ./output/images\Buendia\Buendia - Instruccion_page_005.png -> 351 regions
Processed ./output/images\Buendia\Buendia - Instruccion_page_006.png -> 369 regions
Processed ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_001.png -> 46 regions
Processed ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_002.png -> 76 regions
Processed ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_003.png -> 87 regions
Processed ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_004.png -> 155 regions
Processed ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_005.png -> 43 regions
Processed ./output/images\Constituciones\Constituciones sinodales Calahorra 1602_page_006.png -> 54 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_001.png -> 169 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_002.png -> 125 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_003.png -> 205 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_004.png -> 188 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_005.png -> 223 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_006.png -> 230 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_007.png -> 206 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_008.png -> 215 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_009.png -> 223 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_010.png -> 216 regions
Processed ./output/images\Ezcaray\Ezcaray - Vozes_page_011.png -> 100 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_001.png -> 143 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_002.png -> 89 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_003.png -> 95 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_004.png -> 108 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_005.png -> 91 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_006.png -> 186 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_007.png -> 87 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_008.png -> 185 regions
Processed ./output/images\Mendo\Mendo - Principe perfecto_page_009.png -> 70 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_001.png -> 542 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_002.png -> 615 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_003.png -> 673 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_004.png -> 757 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_005.png -> 689 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_006.png -> 542 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_007.png -> 696 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_008.png -> 689 regions
Processed ./output/images\Paredes\Paredes - Reglas generales_page_009.png -> 775 regions
Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_001.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_002.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_003.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_004.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_005.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_006.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_007.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_008.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_009.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_010.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_011.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_012.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_013.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_014.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_015.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-
python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'
```

```
Error processing ./output/images\PORCONES.228.35\PORCONES.228.35 – 1636_page_016.png: OpenCV(4.11.0) D:\a\opencv-python\opencv-python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'

Document processing complete. Results saved to ./output/regions
```

In [14]:

```python
# 7. Layout Model Evaluation
def evaluate_layout_model(pred_masks_dir, gt_masks_dir, output_file):
    """
    Evaluate layout segmentation model performance.

    Args:
        pred_masks_dir: Directory containing predicted masks
        gt_masks_dir: Directory containing ground truth masks
        output_file: Path to save evaluation results
    """
    # Create output directory
    os.makedirs(os.path.dirname(output_file), exist_ok=True)

    # Get all predicted mask files
    pred_files = []
    for root, _, files in os.walk(pred_masks_dir):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                pred_files.append(os.path.join(root, file))

    # Metrics
    pixel_accuracies = []
    ious = []
    precisions = []
    recalls = []
    f1_scores = []

    # Process each mask
    for pred_path in tqdm(pred_files, desc="Evaluating masks"):
        # Get filename
        filename = os.path.basename(pred_path)

        # Find matching ground truth file
        gt_path = find_matching_gt_file(gt_masks_dir, filename)

        # Skip if ground truth doesn't exist
        if gt_path is None:
            print(f"Warning: Ground truth not found for {filename}, skipping...")
            continue

        # Load masks
        pred_mask = cv2.imread(pred_path, cv2.IMREAD_GRAYSCALE)
        gt_mask = cv2.imread(gt_path, cv2.IMREAD_GRAYSCALE)

        # Ensure same size
        if pred_mask.shape != gt_mask.shape:
            gt_mask = cv2.resize(gt_mask, (pred_mask.shape[1], pred_mask.shape[0]))

        # Binarize masks
        pred_mask = (pred_mask > 127).astype(np.uint8)
        gt_mask = (gt_mask > 127).astype(np.uint8)

        # Calculate metrics
        pixel_accuracy = np.mean(pred_mask == gt_mask)

        # Calculate IoU
        intersection = np.logical_and(pred_mask, gt_mask).sum()
        union = np.logical_or(pred_mask, gt_mask).sum()
        iou = intersection / union if union > 0 else 0.0

        # Calculate precision, recall, and F1
        pred_flat = pred_mask.flatten()
        gt_flat = gt_mask.flatten()

        precision = precision_score(gt_flat, pred_flat, zero_division=0)
        recall = recall_score(gt_flat, pred_flat, zero_division=0)
        f1 = f1_score(gt_flat, pred_flat, zero_division=0)

        # Store metrics
        pixel_accuracies.append(pixel_accuracy)
        ious.append(iou)
        precisions.append(precision)
        recalls.append(recall)
        f1_scores.append(f1)

    # Calculate average metrics
    avg_pixel_accuracy = np.mean(pixel_accuracies)
    avg_iou = np.mean(ious)
    avg_precision = np.mean(precisions)
    avg_recall = np.mean(recalls)
    avg_f1 = np.mean(f1_scores)
```

```
    # Calculate standard deviations
    std_pixel_accuracy = np.std(pixel_accuracies)
    std_iou = np.std(ious)
    std_precision = np.std(precisions)
    std_recall = np.std(recalls)
    std_f1 = np.std(f1_scores)

    # Save results
    with open(output_file, 'w') as f:
        f.write("Layout Segmentation Evaluation Results\n")
        f.write("======================================\n\n")
        f.write(f"Number of evaluated images: {len(pixel_accuracies)}\n\n")

        f.write("Average Metrics:\n")
        f.write("--------------\n")
        f.write(f"Pixel Accuracy: {avg_pixel_accuracy:.4f} ± {std_pixel_accuracy:.4f}\n")
        f.write(f"IoU: {avg_iou:.4f} ± {std_iou:.4f}\n")
        f.write(f"Precision: {avg_precision:.4f} ± {std_precision:.4f}\n")
        f.write(f"Recall: {avg_recall:.4f} ± {std_recall:.4f}\n")
        f.write(f"F1 Score: {avg_f1:.4f} ± {std_f1:.4f}\n")

    # Print summary
    print("\nEvaluation Summary:")
    print("------------------")
    print(f"Pixel Accuracy: {avg_pixel_accuracy:.4f} ± {std_pixel_accuracy:.4f}")
    print(f"IoU: {avg_iou:.4f} ± {std_iou:.4f}")
    print(f"Precision: {avg_precision:.4f} ± {std_precision:.4f}")
    print(f"Recall: {avg_recall:.4f} ± {std_recall:.4f}")
    print(f"F1 Score: {avg_f1:.4f} ± {std_f1:.4f}")
    print(f"\nDetailed results saved to {output_file}")

    return {
        'pixel_accuracy': avg_pixel_accuracy,
        'iou': avg_iou,
        'precision': avg_precision,
        'recall': avg_recall,
        'f1': avg_f1
    }

def find_matching_gt_file(gt_dir, pred_filename):
    """Find a matching ground truth file for a predicted mask filename"""
    # Try exact match first
    exact_path = os.path.join(gt_dir, pred_filename)
    if os.path.exists(exact_path):
        return exact_path

    # Try different extensions
    name_without_ext = os.path.splitext(pred_filename)[0]
    for ext in ['.png', '.jpg', '.jpeg']:
        potential_path = os.path.join(gt_dir, name_without_ext + ext)
        if os.path.exists(potential_path):
            return potential_path

    # Try with mask suffix
    for ext in ['.png', '.jpg', '.jpeg']:
        potential_path = os.path.join(gt_dir, name_without_ext + '_mask' + ext)
        if os.path.exists(potential_path):
            return potential_path

    return None

# Evaluate layout model
layout_eval_results = evaluate_layout_model(
    pred_masks_dir=os.path.join(regions_dir, "masks"),
    gt_masks_dir=masks_dir,
    output_file=os.path.join(base_dir, "output/layout_evaluation.txt")
)
```

```
Evaluating masks:   0%|          | 0/82 [00:00<?, ?it/s]
Evaluation Summary:
------------------
Pixel Accuracy: 0.9083 ± 0.0591
IoU: 0.5751 ± 0.1789
Precision: 0.6641 ± 0.1706
Recall: 0.7872 ± 0.1905
F1 Score: 0.7115 ± 0.1668

Detailed results saved to ./output/layout_evaluation.txt
```

In [ ]: