```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Prerequisites:
# pip install onnxruntime
# pip install onnxruntime-gpu
```

```
!pip install onnxruntime
```

```
Collecting onnxruntime
  Downloading onnxruntime-1.21.0-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (4.5 kB)
Collecting coloredlogs (from onnxruntime)
  Downloading coloredlogs-15.0.1-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: flatbuffers in /usr/local/lib/python3.11/dist-packages (from onnxruntime) (25.2.10)
Requirement already satisfied: numpy>=1.21.6 in /usr/local/lib/python3.11/dist-packages (from onnxruntime) (2.0.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from onnxruntime) (24.2)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from onnxruntime) (5.29.3)
Requirement already satisfied: sympy in /usr/local/lib/python3.11/dist-packages (from onnxruntime) (1.13.1)
Collecting humanfriendly>=9.1 (from coloredlogs->onnxruntime)
  Downloading humanfriendly-10.0-py2.py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->onnxruntime) (1.3.0)
Downloading onnxruntime-1.21.0-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (16.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.0/16.0 MB 36.4 MB/s eta 0:00:00
Downloading coloredlogs-15.0.1-py2.py3-none-any.whl (46 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 3.1 MB/s eta 0:00:00
Downloading humanfriendly-10.0-py2.py3-none-any.whl (86 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 6.0 MB/s eta 0:00:00
Installing collected packages: humanfriendly, coloredlogs, onnxruntime
Successfully installed coloredlogs-15.0.1 humanfriendly-10.0 onnxruntime-1.21.0
```

```
!pip install onnxruntime-gpu
```

```
Collecting onnxruntime-gpu
  Downloading onnxruntime_gpu-1.21.0-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (4.8 kB)
Requirement already satisfied: coloredlogs in /usr/local/lib/python3.11/dist-packages (from onnxruntime-gpu) (15.0.1)
Requirement already satisfied: flatbuffers in /usr/local/lib/python3.11/dist-packages (from onnxruntime-gpu) (25.2.10)
Requirement already satisfied: numpy>=1.21.6 in /usr/local/lib/python3.11/dist-packages (from onnxruntime-gpu) (2.0.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from onnxruntime-gpu) (24.2)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from onnxruntime-gpu) (5.29.3)
Requirement already satisfied: sympy in /usr/local/lib/python3.11/dist-packages (from onnxruntime-gpu) (1.13.1)
Requirement already satisfied: humanfriendly>=9.1 in /usr/local/lib/python3.11/dist-packages (from coloredlogs->onnxruntime-gpu) (10.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->onnxruntime-gpu) (1.3.0)
Downloading onnxruntime_gpu-1.21.0-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (280.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 280.8/280.8 MB 3.2 MB/s eta 0:00:00
Installing collected packages: onnxruntime-gpu
Successfully installed onnxruntime-gpu-1.21.0
```

```
!pip install docling_core
```

```
Collecting docling_core
  Downloading docling_core-2.23.3-py3-none-any.whl.metadata (5.8 kB)
Collecting jsonref<2.0.0,>=1.1.0 (from docling_core)
  Downloading jsonref-1.1.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: jsonschema<5.0.0,>=4.16.0 in /usr/local/lib/python3.11/dist-packages (from docling_core) (4.23.0)
Collecting latex2mathml<4.0.0,>=3.77.0 (from docling_core)
  Downloading latex2mathml-3.77.0-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: pandas<3.0.0,>=2.1.4 in /usr/local/lib/python3.11/dist-packages (from docling_core) (2.2.2)
Requirement already satisfied: pillow<12.0.0,>=10.0.0 in /usr/local/lib/python3.11/dist-packages (from docling_core) (11.1.0)
Requirement already satisfied: pydantic!=2.10.0,!=2.10.1,!=2.10.2,<3.0.0,>=2.6.0 in /usr/local/lib/python3.11/dist-packages (from doclin
Requirement already satisfied: pyyaml<7.0.0,>=5.1 in /usr/local/lib/python3.11/dist-packages (from docling_core) (6.0.2)
Requirement already satisfied: tabulate<0.10.0,>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from docling_core) (0.9.0)
Collecting typer<0.13.0,>=0.12.5 (from docling_core)
  Downloading typer-0.12.5-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: typing-extensions<5.0.0,>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from docling_core) (4.12.2)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema<5.0.0,>=4.16.0->docling_core) (
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema<5.0.0,>=
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema<5.0.0,>=4.16.0->docling_c
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema<5.0.0,>=4.16.0->docling_core)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0.0,>=2.1.4->docling_core) (2.0.2
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0.0,>=2.1.4->docling_cor
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0.0,>=2.1.4->docling_core) (2025.1
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0.0,>=2.1.4->docling_core) (2025
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=2.10.0,!=2.10.1,!=2.10.
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=2.10.0,!=2.10.1,!=2.10.2
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<0.13.0,>=0.12.5->docling_core) (8.1.8
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<0.13.0,>=0.12.5->docling_core)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<0.13.0,>=0.12.5->docling_core) (13.9
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0.0,>=2.1.4->d
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<0.13.0,>=0.12
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<0.13.0,>=0.
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<0
Downloading docling_core-2.23.3-py3-none-any.whl (115 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.9/115.9 kB 5.2 MB/s eta 0:00:00
Downloading jsonref-1.1.0-py3-none-any.whl (9.4 kB)
Downloading latex2mathml-3.77.0-py3-none-any.whl (73 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.7/73.7 kB 4.9 MB/s eta 0:00:00
Downloading typer-0.12.5-py3-none-any.whl (47 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.3/47.3 kB 2.6 MB/s eta 0:00:00
Installing collected packages: latex2mathml, jsonref, typer, docling_core
  Attempting uninstall: typer
    Found existing installation: typer 0.15.2
    Uninstalling typer-0.15.2:
      Successfully uninstalled typer-0.15.2
Successfully installed docling_core-2.23.3 jsonref-1.1.0 latex2mathml-3.77.0 typer-0.12.5
```

```python

from transformers import AutoConfig, AutoProcessor
from transformers.image_utils import load_image
import onnxruntime
import numpy as np
import os
from docling_core.types.doc import DoclingDocument
from docling_core.types.doc.document import DocTagsDocument

os.environ["OMP_NUM_THREADS"] = "1"
# cuda
os.environ["ORT_CUDA_USE_MAX_WORKSPACE"] = "1"

# 1. Load models
## Load config and processor
model_id = "ds4sd/SmolDocling-256M-preview"
config = AutoConfig.from_pretrained(model_id)
processor = AutoProcessor.from_pretrained(model_id)

## Load sessions
# !wget https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/vision_encoder.onnx
# !wget https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/embed_tokens.onnx
# !wget https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/decoder_model_merged.onnx
# cpu
# vision_session = onnxruntime.InferenceSession("vision_encoder.onnx")
# embed_session = onnxruntime.InferenceSession("embed_tokens.onnx")
# decoder_session = onnxruntime.InferenceSession("decoder_model_merged.onnx"

# cuda
vision_session = onnxruntime.InferenceSession("vision_encoder.onnx", providers=["CUDAExecutionProvider"])
embed_session = onnxruntime.InferenceSession("embed_tokens.onnx", providers=["CUDAExecutionProvider"])
decoder_session = onnxruntime.InferenceSession("decoder_model_merged.onnx", providers=["CUDAExecutionProvider"])

## Set config values
num_key_value_heads = config.text_config.num_key_value_heads
head_dim = config.text_config.head_dim
num_hidden_layers = config.text_config.num_hidden_layers
eos_token_id = config.text_config.eos_token_id
image_token_id = config.image_token_id
end_of_utterance_id = processor.tokenizer.convert_tokens_to_ids("<end_of_utterance>")

# 2. Prepare inputs
## Create input messages
messages = [
    {
        "role": "user",
        "content": [
            {"type": "image"},
            {"type": "text", "text": "Convert this page to docling."}
        ]
    },
]

## Load image and apply processor
image = load_image("https://ibm.biz/docling-page-with-table")
prompt = processor.apply_chat_template(messages, add_generation_prompt=True)
inputs = processor(text=prompt, images=[image], return_tensors="np")

## Prepare decoder inputs
```

```
batch_size = inputs['input_ids'].shape[0]
past_key_values = {
    f'past_key_values.{layer}.{kv}': np.zeros([batch_size, num_key_value_heads, 0, head_dim], dtype=np.float32)
    for layer in range(num_hidden_layers)
    for kv in ('key', 'value')
}
image_features = None
input_ids = inputs['input_ids']
attention_mask = inputs['attention_mask']
position_ids = np.cumsum(inputs['attention_mask'], axis=-1)


# 3. Generation loop
max_new_tokens = 8192
generated_tokens = np.array([[]], dtype=np.int64)
for i in range(max_new_tokens):
  inputs_embeds = embed_session.run(None, {'input_ids': input_ids})[0]

  if image_features is None:
    ## Only compute vision features if not already computed
    image_features = vision_session.run(
        ['image_features'],  # List of output names or indices
        {
            'pixel_values': inputs['pixel_values'],
            'pixel_attention_mask': inputs['pixel_attention_mask'].astype(np.bool_)
        }
    )[0]

    ## Merge text and vision embeddings
    inputs_embeds[inputs['input_ids'] == image_token_id] = image_features.reshape(-1, image_features.shape[-1])

  logits, *present_key_values = decoder_session.run(None, dict(
      inputs_embeds=inputs_embeds,
      attention_mask=attention_mask,
      position_ids=position_ids,
      **past_key_values,
  ))

  ## Update values for next generation loop
  input_ids = logits[:, -1].argmax(-1, keepdims=True)
  attention_mask = np.ones_like(input_ids)
  position_ids = position_ids[:, -1:] + 1
  for j, key in enumerate(past_key_values):
    past_key_values[key] = present_key_values[j]

  generated_tokens = np.concatenate([generated_tokens, input_ids], axis=-1)
  if (input_ids == eos_token_id).all() or (input_ids == end_of_utterance_id).all():
    break  # Stop predicting

doctags = processor.batch_decode(
    generated_tokens,
    skip_special_tokens=False,
)[0].lstrip()

print(doctags)

doctags_doc = DocTagsDocument.from_doctags_and_image_pairs([doctags], [image])
print(doctags)
# create a docling document
doc = DoclingDocument(name="Document")
doc.load_from_doctags(doctags_doc)

print(doc.export_to_markdown())
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%                                                 3.90k/3.90k [00:00<00:00, 314kB/s]
processor_config.json: 100%                                         68.0/68.0 [00:00<00:00, 4.17kB/s]
chat_template.json: 100%                                          430/430 [00:00<00:00, 31.6kB/s]
preprocessor_config.json: 100%                                      486/486 [00:00<00:00, 28.7kB/s]
tokenizer_config.json: 100%                                       27.4k/27.4k [00:00<00:00, 1.13MB/s]
vocab.json: 100%                                                  801k/801k [00:00<00:00, 9.12MB/s]
merges.txt: 100%                                                  466k/466k [00:00<00:00, 6.28MB/s]
tokenizer.json: 100%                                              3.55M/3.55M [00:00<00:00, 9.68MB/s]
added_tokens.json: 100%                                           3.67k/3.67k [00:00<00:00, 314kB/s]
special_tokens_map.json: 100%                                       1.07k/1.07k [00:00<00:00, 117kB/s]
---------------------------------------------------------------------------
NoSuchFile                                Traceback (most recent call last)
<ipython-input-5-251b49be7d7b> in <cell line: 0>()
     27
     28 # cuda
---> 29 vision_session = onnxruntime.InferenceSession("vision_encoder.onnx", providers=["CUDAExecutionProvider"])
     30 embed_session = onnxruntime.InferenceSession("embed_tokens.onnx", providers=["CUDAExecutionProvider"])
     31 decoder_session = onnxruntime.InferenceSession("decoder_model_merged.onnx", providers=["CUDAExecutionProvider"])

                                  1 frames
/usr/local/lib/python3.11/dist-packages/onnxruntime/capi/onnxruntime_inference_collection.py in _create_inference_session(self,
providers, provider_options, disabled_optimizers)
    528
    529         if self._model_path:
--> 530             sess = C.InferenceSession(session_options, self._model_path, True, self._read_config_from_model)
    531         else:
    532             sess = C.InferenceSession(session_options, self._model_bytes, False, self._read_config_from_model)

NoSuchFile: [ONNXRuntimeError] : 3 : NO_SUCHFILE : Load model from vision_encoder.onnx failed:Load model vision_encoder.onnx failed.
File doesn't exist
```

Next steps: Explain error

```python
from transformers import AutoConfig, AutoProcessor
from transformers.image_utils import load_image
import onnxruntime
import numpy as np
import os
from docling_core.types.doc import DoclingDocument
from docling_core.types.doc.document import DocTagsDocument

os.environ["OMP_NUM_THREADS"] = "1"
# cuda
os.environ["ORT_CUDA_USE_MAX_WORKSPACE"] = "1"

# 1. Load models
## Load config and processor
model_id = "ds4sd/SmolDocling-256M-preview"
config = AutoConfig.from_pretrained(model_id)
processor = AutoProcessor.from_pretrained(model_id)

## Load sessions
# Download the ONNX models if they are not present
!wget https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/vision_encoder.onnx
!wget https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/embed_tokens.onnx
!wget https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/decoder_model_merged.onnx
# cpu
# vision_session = onnxruntime.InferenceSession("vision_encoder.onnx")
# embed_session = onnxruntime.InferenceSession("embed_tokens.onnx")
# decoder_session = onnxruntime.InferenceSession("decoder_model_merged.onnx"

# cuda
```

```
vision_session = onnxruntime.InferenceSession("vision_encoder.onnx", providers=["CUDAExecutionProvider"])
embed_session = onnxruntime.InferenceSession("embed_tokens.onnx", providers=["CUDAExecutionProvider"])
decoder_session = onnxruntime.InferenceSession("decoder_model_merged.onnx", providers=["CUDAExecutionProvider"])

# ... rest of the code ...
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
--2025-03-23 00:18:28--  https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/vision_encoder.onnx
Resolving huggingface.co (huggingface.co)... 18.164.174.17, 18.164.174.118, 18.164.174.55, ...
Connecting to huggingface.co (huggingface.co)|18.164.174.17|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://cdn-lfs-us-1.hf.co/repos/b8/6e/b86e88e6db456aab3c09c424eee0c6e3c444c4122486d3f9ec088b13c389b0f9/9c97aafac0543632c77893
--2025-03-23 00:18:28--  https://cdn-lfs-us-1.hf.co/repos/b8/6e/b86e88e6db456aab3c09c424eee0c6e3c444c4122486d3f9ec088b13c389b0f9/9c97aaf
Resolving cdn-lfs-us-1.hf.co (cdn-lfs-us-1.hf.co)... 3.167.192.98, 3.167.192.6, 3.167.192.118, ...
Connecting to cdn-lfs-us-1.hf.co (cdn-lfs-us-1.hf.co)|3.167.192.98|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 374316454 (357M) [binary/octet-stream]
Saving to: ‘vision_encoder.onnx’

vision_encoder.onnx 100%[===================>] 356.98M   149MB/s    in 2.4s

2025-03-23 00:18:31 (149 MB/s) - ‘vision_encoder.onnx’ saved [374316454/374316454]

--2025-03-23 00:18:31--  https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/embed_tokens.onnx
Resolving huggingface.co (huggingface.co)... 18.164.174.17, 18.164.174.118, 18.164.174.55, ...
Connecting to huggingface.co (huggingface.co)|18.164.174.17|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://cdn-lfs-us-1.hf.co/repos/b8/6e/b86e88e6db456aab3c09c424eee0c6e3c444c4122486d3f9ec088b13c389b0f9/816939a02a48a540330655
--2025-03-23 00:18:31--  https://cdn-lfs-us-1.hf.co/repos/b8/6e/b86e88e6db456aab3c09c424eee0c6e3c444c4122486d3f9ec088b13c389b0f9/816939a
Resolving cdn-lfs-us-1.hf.co (cdn-lfs-us-1.hf.co)... 3.167.192.98, 3.167.192.6, 3.167.192.118, ...
Connecting to cdn-lfs-us-1.hf.co (cdn-lfs-us-1.hf.co)|3.167.192.98|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 113541419 (108M) [binary/octet-stream]
Saving to: ‘embed_tokens.onnx’

embed_tokens.onnx   100%[===================>] 108.28M   121MB/s    in 0.9s

2025-03-23 00:18:32 (121 MB/s) - ‘embed_tokens.onnx’ saved [113541419/113541419]

--2025-03-23 00:18:32--  https://huggingface.co/ds4sd/SmolDocling-256M-preview/resolve/main/onnx/decoder_model_merged.onnx
Resolving huggingface.co (huggingface.co)... 18.164.174.17, 18.164.174.118, 18.164.174.55, ...
Connecting to huggingface.co (huggingface.co)|18.164.174.17|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://cdn-lfs-us-1.hf.co/repos/b8/6e/b86e88e6db456aab3c09c424eee0c6e3c444c4122486d3f9ec088b13c389b0f9/c75155e5e121b9d89bf543
--2025-03-23 00:18:32--  https://cdn-lfs-us-1.hf.co/repos/b8/6e/b86e88e6db456aab3c09c424eee0c6e3c444c4122486d3f9ec088b13c389b0f9/c75155e
Resolving cdn-lfs-us-1.hf.co (cdn-lfs-us-1.hf.co)... 3.167.192.98, 3.167.192.6, 3.167.192.118, ...
Connecting to cdn-lfs-us-1.hf.co (cdn-lfs-us-1.hf.co)|3.167.192.98|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 540972236 (516M) [binary/octet-stream]
Saving to: ‘decoder_model_merged.onnx’

decoder_model_merge 100%[===================>] 515.91M   165MB/s    in 3.2s

2025-03-23 00:18:35 (160 MB/s) - ‘decoder_model_merged.onnx’ saved [540972236/540972236]
```

```
## Set config values
num_key_value_heads = config.text_config.num_key_value_heads
head_dim = config.text_config.head_dim
num_hidden_layers = config.text_config.num_hidden_layers
eos_token_id = config.text_config.eos_token_id
image_token_id = config.image_token_id
end_of_utterance_id = processor.tokenizer.convert_tokens_to_ids("<end_of_utterance>")

# 2. Prepare inputs
## Create input messages
messages = [
    {
        "role": "user",
        "content": [
            {"type": "image"},
            {"type": "text", "text": "Convert this page to docling."}
        ]
    },
]
```

```

## Load image and apply processor
image = load_image("https://ibm.biz/docling-page-with-table")
prompt = processor.apply_chat_template(messages, add_generation_prompt=True)
inputs = processor(text=prompt, images=[image], return_tensors="np")

## Prepare decoder inputs
batch_size = inputs['input_ids'].shape[0]
past_key_values = {
    f'past_key_values.{layer}.{kv}': np.zeros([batch_size, num_key_value_heads, 0, head_dim], dtype=np.float32)
    for layer in range(num_hidden_layers)
    for kv in ('key', 'value')
}
image_features = None
input_ids = inputs['input_ids']
attention_mask = inputs['attention_mask']
position_ids = np.cumsum(inputs['attention_mask'], axis=-1)


# 3. Generation loop
max_new_tokens = 8192
generated_tokens = np.array([[]], dtype=np.int64)
for i in range(max_new_tokens):
  inputs_embeds = embed_session.run(None, {'input_ids': input_ids})[0]

  if image_features is None:
    ## Only compute vision features if not already computed
    image_features = vision_session.run(
        ['image_features'],  # List of output names or indices
        {
            'pixel_values': inputs['pixel_values'],
            'pixel_attention_mask': inputs['pixel_attention_mask'].astype(np.bool_)
        }
    )[0]

    ## Merge text and vision embeddings
    inputs_embeds[inputs['input_ids'] == image_token_id] = image_features.reshape(-1, image_features.shape[-1])

  logits, *present_key_values = decoder_session.run(None, dict(
      inputs_embeds=inputs_embeds,
      attention_mask=attention_mask,
      position_ids=position_ids,
      **past_key_values,
  ))

  ## Update values for next generation loop
  input_ids = logits[:, -1].argmax(-1, keepdims=True)
  attention_mask = np.ones_like(input_ids)
  position_ids = position_ids[:, -1:] + 1
  for j, key in enumerate(past_key_values):
    past_key_values[key] = present_key_values[j]

  generated_tokens = np.concatenate([generated_tokens, input_ids], axis=-1)
  if (input_ids == eos_token_id).all() or (input_ids == end_of_utterance_id).all():
    break  # Stop predicting

doctags = processor.batch_decode(
    generated_tokens,
    skip_special_tokens=False,
)[0].lstrip()

print(doctags)

doctags_doc = DocTagsDocument.from_doctags_and_image_pairs([doctags], [image])
print(doctags)
# create a docling document
doc = DoclingDocument(name="Document")
doc.load_from_doctags(doctags_doc)

print(doc.export_to_markdown())
```

```
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly trunca
<doctag><page_header><loc_127><loc_27><loc_420><loc_34>Optimized Table Tokenization for Table Structure Recognition</page_header>
<page_header><loc_451><loc_27><loc_457><loc_34>9</page_header>
<text><loc_57><loc_46><loc_457><loc_71>order to compute the TED score. Inference timing results for all experiments were obtained from t
<section_header_level_1><loc_57><loc_84><loc_271><loc_92>5.1 Hyper Parameter Optimization</section_header_level_1>
<text><loc_57><loc_97><loc_457><loc_149>We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of
<otsl><loc_62><loc_212><loc_451><loc_315><ched>#<ched>#<ched>#<ched>TEDs<lcel><ched>mAP<ched>Inference  time (secs)<nl><rhed>enc-layers<
```

<section_header_level_1><loc_57><loc_343><loc_209><loc_351>5.2 Quantitative Results</section_header_level_1>
<text><loc_57><loc_356><loc_457><loc_427>We picked the model parameter configuration that produced the best prediction quality (enc=6, d
<text><loc_57><loc_429><loc_457><loc_464>Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger da
</doctag><end_of_utterance>
<doctag><page_header><loc_127><loc_27><loc_420><loc_34>Optimized Table Tokenization for Table Structure Recognition</page_header>
<page_header><loc_451><loc_27><loc_457><loc_34>9</page_header>
<text><loc_57><loc_46><loc_457><loc_71>order to compute the TED score. Inference timing results for all experiments were obtained from t
<section_header_level_1><loc_57><loc_84><loc_271><loc_92>5.1 Hyper Parameter Optimization</section_header_level_1>
<text><loc_57><loc_97><loc_457><loc_149>We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of
<otsl><loc_62><loc_212><loc_451><loc_315><ched>#<ched>#<ched>#<ched>TEDs<lcel><ched>mAP<ched>Inference time (secs)<nl><rhed>enc-layers<
<section_header_level_1><loc_57><loc_343><loc_209><loc_351>5.2 Quantitative Results</section_header_level_1>
<text><loc_57><loc_356><loc_457><loc_427>We picked the model parameter configuration that produced the best prediction quality (enc=6, d
<text><loc_57><loc_429><loc_457><loc_464>Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger da
</doctag><end_of_utterance>

Optimized Table Tokenization for Table Structure Recognition

9

order to compute the TED score. Inference timing results for all experiments were obtained from the same machine on a single core with A

5.1 Hyper Parameter Optimization

We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of tables. Also we report TED scores separa

Table 1. HPO performed in OTSL and HTML representation on the same transformer-based TableFormer [9] architecture, trained only on PubTa

#	#	#	TEDs	TEDs	mAP	Inference time (secs)	
enc-layers	dec-layers	Language	simple	complex	all	(0.75)	time (secs)
6	6	OTSL	0.965	0.934	0.955	0.88	2.73
		HTML	0.969	0.927	0.955	0.857	5.39
4	4	OTSL	0.938	0.904	0.927	0.853	1.97
		HTML	0.952	0.909	0.938	0.843	3.77
2	4	OTSL	0.923	0.897	0.915	0.859	1.91
		HTML	0.945	0.901	0.931	0.834	3.81
4	2	OTSL	0.952	0.92	0.942	0.857	1.22
		HTML	0.944	0.903	0.931	0.824	2

5.2 Quantitative Results

We picked the model parameter configuration that produced the best prediction quality (enc=6, dec=6, heads=8) with PubTabNet alone, then

Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger data set like PubTables-1M and achieves sig

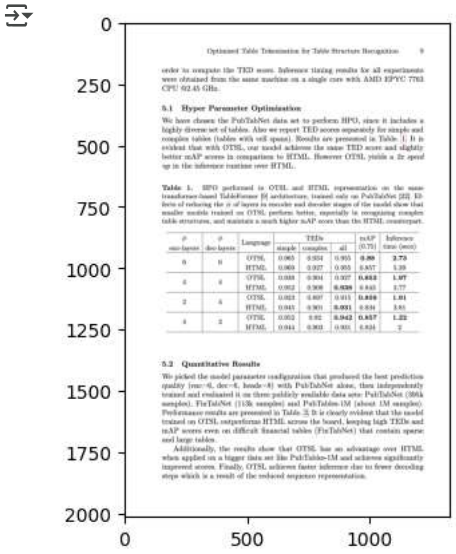
```
Start coding or generate with AI.
```

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# مسار الصورة
image_path = "/content/page_with_table.png"

# قراءة الصورة
img = mpimg.imread(image_path)

# عرض الصورة
imgplot = plt.imshow(img)
plt.show()
```



Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

```python
# Prerequisites:
# pip install torch
# pip install docling_core
# pip install transformers

import torch
from docling_core.types.doc import DoclingDocument
from docling_core.types.doc.document import DocTagsDocument
from transformers import AutoProcessor, AutoModelForVision2Seq
from transformers.image_utils import load_image
from pathlib import Path

DEVICE = "cuda" if torch.cuda.is_available() else "cpu"

# Load images
image = load_image("https://upload.wikimedia.org/wikipedia/commons/7/76/GazettedeFrance.jpg")

# Initialize processor and model
processor = AutoProcessor.from_pretrained("ds4sd/SmolDocling-256M-preview")
model = AutoModelForVision2Seq.from_pretrained(
    "ds4sd/SmolDocling-256M-preview",
    torch_dtype=torch.bfloat16,
    _attn_implementation="flash_attention_2" if DEVICE == "cuda" else "eager",
).to(DEVICE)

# Create input messages
messages = [
    {
        "role": "user",
        "content": [
            {"type": "image"},
            {"type": "text", "text": "Convert this page to docling."}
        ]
    },
]

# Prepare inputs
```

```
prompt = processor.apply_chat_template(messages, add_generation_prompt=True)
inputs = processor(text=prompt, images=[image], return_tensors="pt")
inputs = inputs.to(DEVICE)

# Generate outputs
generated_ids = model.generate(**inputs, max_new_tokens=8192)
prompt_length = inputs.input_ids.shape[1]
trimmed_generated_ids = generated_ids[:, prompt_length:]
doctags = processor.batch_decode(
    trimmed_generated_ids,
    skip_special_tokens=False,
)[0].lstrip()

# Populate document
```