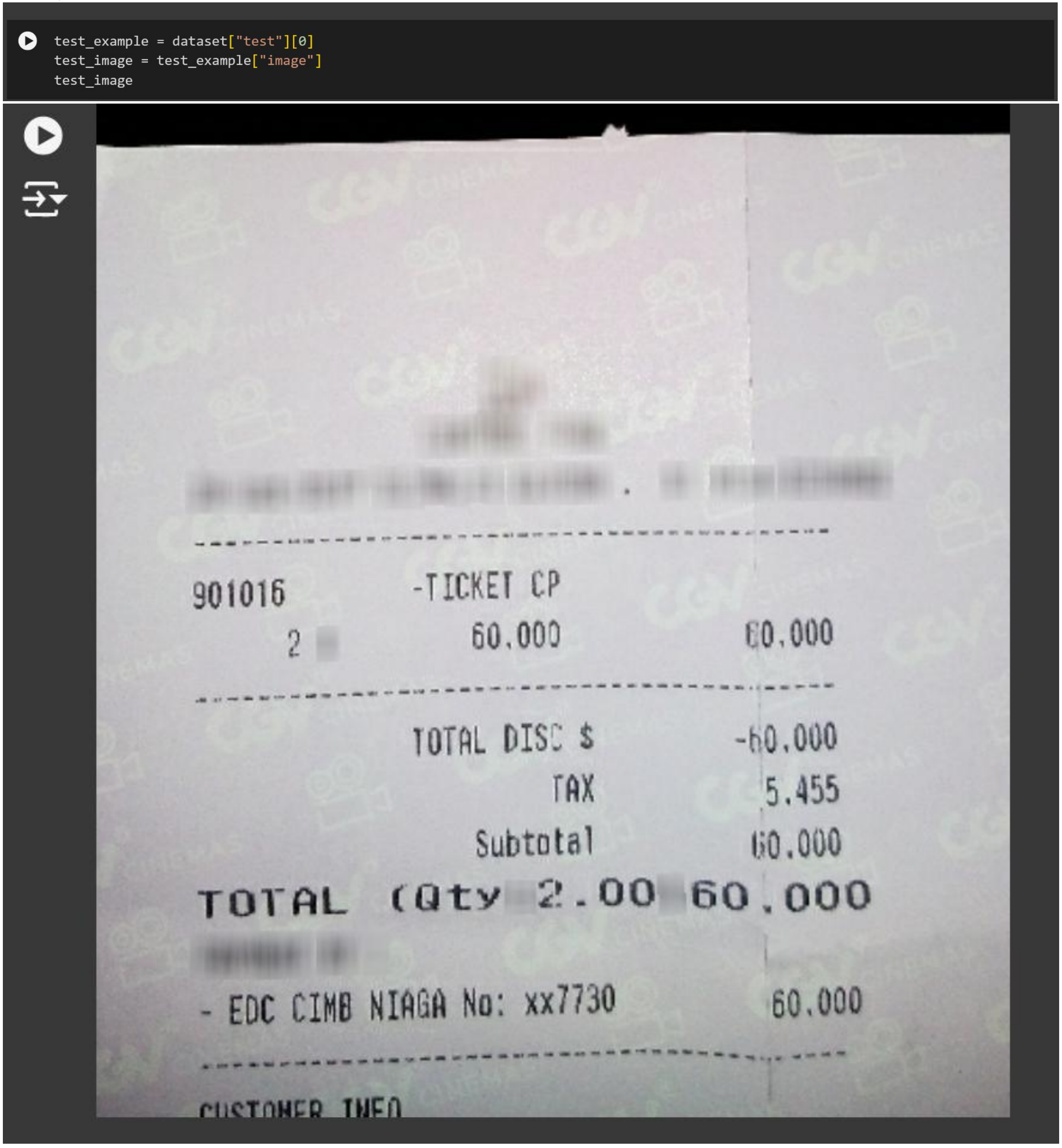
In this notebook, I have shown the process of fine-tuning Google's recent vision model PaliGemma on a dataset provided by Dount which has images of bills and ground truth given as data.

I have fine-tuned the model with limited resources available on google colab and the model performs well on the given data after fine-tuning.

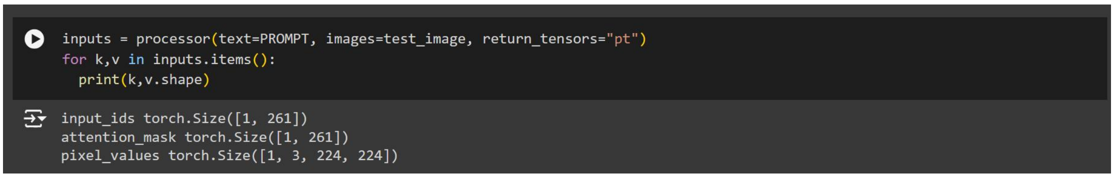
Sample from test dataset:



```
test_example = dataset["test"][0]
test_image = test_example["image"]
test_image
```

```
[redacted]
------------------------------------------------
901016            -TICKET CP
     2 [redacted]       60.000               60.000
------------------------------------------------
                  TOTAL DISC $          -60.000
                           TAX            5.455
                      Subtotal           60.000
TOTAL (Qty [redacted] 2.00 [redacted] 60.000
[redacted]
- EDC CIMB NIAGA No: xx7730              60.000
------------------------------------------------
CUSTOMER INFO
```

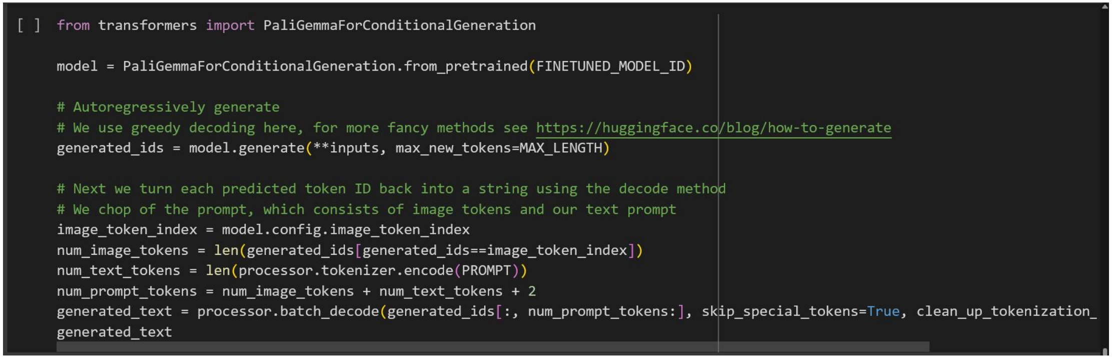
Given this image as an input, the model tries to predict the ground truth from the text.



```
inputs = processor(text=PROMPT, images=test_image, return_tensors="pt")
for k,v in inputs.items():
  print(k,v.shape)
```

```
input_ids torch.Size([1, 261])
attention_mask torch.Size([1, 261])
pixel_values torch.Size([1, 3, 224, 224])
```

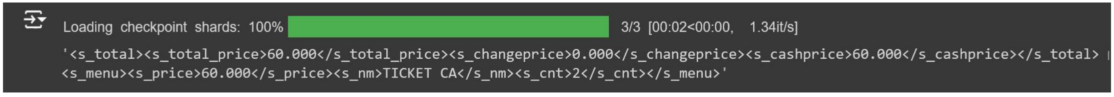
Passing the image to the tokenizer to get tokens to pass as inputs to the model



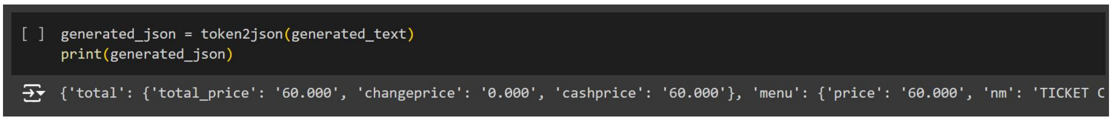
```
from transformers import PaliGemmaForConditionalGeneration

model = PaliGemmaForConditionalGeneration.from_pretrained(FINETUNED_MODEL_ID)

# Autoregressively generate
# We use greedy decoding here, for more fancy methods see https://huggingface.co/blog/how-to-generate
generated_ids = model.generate(**inputs, max_new_tokens=MAX_LENGTH)

# Next we turn each predicted token ID back into a string using the decode method
# We chop of the prompt, which consists of image tokens and our text prompt
image_token_index = model.config.image_token_index
num_image_tokens = len(generated_ids[generated_ids==image_token_index])
num_text_tokens = len(processor.tokenizer.encode(PROMPT))
num_prompt_tokens = num_image_tokens + num_text_tokens + 2
generated_text = processor.batch_decode(generated_ids[:, num_prompt_tokens:], skip_special_tokens=True, clean_up_tokenization_
generated_text
```

Loading the Fine-tuned model from the huggingface repo and using generate method to get the predicted tokens.
Using processor(tokenizer) to change predicted tokens into text.

Output:



```
Loading checkpoint shards: 100% 3/3 [00:02<00:00,  1.34it/s]
'<s_total><s_total_price>60.000</s_total_price><s_changeprice>0.000</s_changeprice><s_cashprice>60.000</s_cashprice></s_total>
<s_menu><s_price>60.000</s_price><s_nm>TICKET CA</s_nm><s_cnt>2</s_cnt></s_menu>'
```



```
generated_json = token2json(generated_text)
print(generated_json)
```

```
{'total': {'total_price': '60.000', 'changeprice': '0.000', 'cashprice': '60.000'}, 'menu': {'price': '60.000', 'nm': 'TICKET C
```

Converting the text back into json.

Satyam Rai
9004145893