Here’s a table of available **Hugging Face models** for each category based on your requirements:

|  |  |  |
| --- | --- | --- |
| **Category** | **Model Name** | **Description** |
| **Question Answering** | deepset/roberta-base-squad2 | RoBERTa fine-tuned on SQuAD v2.0 |
|  | bert-large-uncased-whole-word-masking-finetuned-squad | BERT fine-tuned on SQuAD v1.1 |
|  | distilbert-base-cased-distilled-squad | DistilBERT distilled for question answering |
| **Code Generation** | Salesforce/codegen-350M-mono | Code generation model trained on Python |
|  | Salesforce/codegen-2B-mono | Larger CodeGen model (2B parameters) |
|  | bigcode/starcoder | StarCoder trained on multiple programming languages |
| **Summarization** | facebook/bart-large-cnn | BART model trained for text summarization |
|  | t5-small | T5 small model for abstractive summarization |
|  | google/pegasus-xsum | Pegasus trained on the XSum dataset |
| **Translation** | Helsinki-NLP/opus-mt-en-fr | English-to-French translation |
|  | facebook/m2m100\_418M | Multilingual translation model |
|  | t5-base | Supports various NLP tasks, including translation |
| **Text Generation** | gpt2 | OpenAI GPT-2 for text generation |
|  | EleutherAI/gpt-neo-1.3B | GPT-Neo (1.3B parameters) |
|  | mistralai/Mistral-7B-Instruct-v0.1 | Mistral AI's 7B instruct-tuned model |
| **Text Classification** | distilbert-base-uncased-finetuned-sst-2-english | DistilBERT for sentiment classification |
|  | facebook/bart-large-mnli | BART fine-tuned for multi-class classification |
|  | roberta-large-mnli | RoBERTa fine-tuned for natural language inference |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |
| --- | --- |
|  | import gradio as gr |
|  | from transformers import pipeline |
|  | import spaces |
|  | import torch |
|  |  |
|  | zero = torch.Tensor([0]).cuda() |
|  | print(zero.device) |
|  |  |
|  |  |
|  | *# Check if GPU is available for FP16 inference* |
|  | device = 0 if torch.cuda.is\_available() else -1 |
|  | torch\_dtype = torch.float16 if torch.cuda.is\_available() else torch.float32 |
|  |  |
|  | *# Load Pipelines with FP16 (if GPU available)* |
|  | question\_answering = pipeline("question-answering", model="deepset/roberta-base-squad2", device=device) |
|  | code\_generation = pipeline("text-generation", model="Salesforce/codegen-350M-mono", device=device) |
|  | summarization = pipeline("summarization", model="facebook/bart-large-cnn", device=device) |
|  | *#translation = pipeline("translation\_en\_to\_fr", model="Helsinki-NLP/opus-mt-en-fr", device=device)* |
|  | *#translation = pipeline("translation", model="facebook/m2m100\_418M", device=device)* |
|  | text\_generation = pipeline("text-generation", model="gpt2", device=device) |
|  | text\_classification = pipeline("text-classification", model="distilbert-base-uncased-finetuned-sst-2-english", device=device) |
|  |  |
|  | *# Define Functions for Each Task* |
|  | @spaces.GPU |
|  | def answer\_question(context, question): |
|  |  result = question\_answering(question=question, context=context) |
|  |  return result["answer"] |
|  |  |
|  | @spaces.GPU |
|  | def generate\_code(prompt): |
|  |  output = code\_generation(prompt, max\_length=50) |
|  |  return output[0]['generated\_text'] |
|  |  |
|  | @spaces.GPU |
|  | def summarize\_text(text): |
|  |  output = summarization(text, max\_length=100, min\_length=30, do\_sample=False) |
|  |  return output[0]['summary\_text'] |
|  |  |
|  |  |
|  |  |
|  | @spaces.GPU |
|  | def generate\_text(prompt): |
|  |  output = text\_generation(prompt, max\_length=100) |
|  |  return output[0]['generated\_text'] |
|  |  |
|  | @spaces.GPU |
|  | def classify\_text(text): |
|  |  output = text\_classification(text) |
|  |  return f"Label: {output[0]['label']} | Score: {output[0]['score']:.4f}" |
|  |  |
|  | *# Gradio Interface* |
|  | with gr.Blocks() as demo: |
|  |  gr.Markdown("# 🤖 Transformers Pipeline with FP16 Inference") |
|  |   |
|  |  with gr.Tab("1️⃣ Question Answering"): |
|  |  with gr.Row(): |
|  |  context = gr.Textbox(label="Context", lines=4, placeholder="Paste your paragraph here...") |
|  |  question = gr.Textbox(label="Question", placeholder="Ask a question...") |
|  |  answer\_btn = gr.Button("Get Answer") |
|  |  answer\_output = gr.Textbox(label="Answer") |
|  |  answer\_btn.click(answer\_question, inputs=[context, question], outputs=answer\_output) |
|  |  |
|  |  with gr.Tab("2️⃣ Code Generation"): |
|  |  code\_input = gr.Textbox(label="Code Prompt", placeholder="Write code snippet...") |
|  |  code\_btn = gr.Button("Generate Code") |
|  |  code\_output = gr.Textbox(label="Generated Code") |
|  |  code\_btn.click(generate\_code, inputs=code\_input, outputs=code\_output) |
|  |  |
|  |  with gr.Tab("3️⃣ Summarization"): |
|  |  summary\_input = gr.Textbox(label="Text to Summarize", lines=5, placeholder="Paste long text here...") |
|  |  summary\_btn = gr.Button("Summarize") |
|  |  summary\_output = gr.Textbox(label="Summary") |
|  |  summary\_btn.click(summarize\_text, inputs=summary\_input, outputs=summary\_output) |
|  |  |
|  |  with gr.Tab("4️⃣ Text Generation"): |
|  |  text\_input = gr.Textbox(label="Text Prompt", placeholder="Start your text...") |
|  |  text\_btn = gr.Button("Generate Text") |
|  |  text\_output = gr.Textbox(label="Generated Text") |
|  |  text\_btn.click(generate\_text, inputs=text\_input, outputs=text\_output) |
|  |  |
|  |  with gr.Tab("5️⃣ Text Classification"): |
|  |  classify\_input = gr.Textbox(label="Enter Text", placeholder="Enter a sentence...") |
|  |  classify\_btn = gr.Button("Classify Sentiment") |
|  |  classify\_output = gr.Textbox(label="Classification Result") |
|  |  classify\_btn.click(classify\_text, inputs=classify\_input, outputs=classify\_output) |
|  |  |
|  | *# Launch App* |
|  | demo.launch() |
|  |  |

 |