```
from PIL import Image
from termcolor import colored
from transformers import (
    AutoTokenizer,
    FuyuForCausalLM,
    FuyuImageProcessor,
    FuyuProcessor,
)

from swarm_models.base_multimodal_model import BaseMultiModalModel


class Fuyu(BaseMultiModalModel):
    """
    Fuyu model by Adept


    Args:
        BaseMultiModalModel (BaseMultiModalModel): [description]
        model_name (str, optional): [description]. Defaults to "adept/fuyu-8b".
        device_map (str, optional): [description]. Defaults to "auto".
        max_new_tokens (int, optional): [description]. Defaults to 500.
        *args: [description]
        **kwargs: [description]
```

```
    Examples:
    >>> from swarm_models import Fuyu
    >>> model = Fuyu()
    >>> model.run("Hello, world!", "https://upload.wikimedia.org/wikipedia/commons/8/86/Id%C3%A9fix.JPG")
    """

    def __init__(
        self,
        model_name: str = "adept/fuyu-8b",
        device_map: str = "auto",
        max_new_tokens: int = 500,
        *args,
        **kwargs,
    ):
        super().__init__(model_name=model_name, *args, **kwargs)
        self.model_name = model_name
        self.device_map = device_map
        self.max_new_tokens = max_new_tokens

        self.tokenizer = AutoTokenizer.from_pretrained(model_name)
        self.image_processor = FuyuImageProcessor()
        self.processor = FuyuProcessor(
            image_processor=self.image_processor,
            tokenizer=self.tokenizer,
```

```
    )
    self.model = FuyuForCausalLM.from_pretrained(
        model_name,
        device_map=device_map,
        *args,
        **kwargs,
    )

def get_img(self, img: str):
    """Get the image from the path"""
    image_pil = Image.open(img)
    return image_pil

def run(self, text: str = None, img: str = None, *args, **kwargs):
    """Run the pipeline

    Args:
        text (str): _description_
        img (str): _description_

    Returns:
        _type_: _description_
    """
    try:
        img = self.get_img(img)
        model_inputs = self.processor(
```

```
            text=text,
            images=[img],
            device=self.device_map,
        )

        for k, v in model_inputs.items():
            model_inputs[k] = v.to(self.device_map)

        output = self.model.generate(
            max_new_tokens=self.max_new_tokens,
            *args,
            **model_inputs,
            **kwargs,
        )
        text = self.processor.batch_decode(
            output[:, -7:],
            skip_special_tokens=True,
        )
        return print(str(text))
    except Exception as error:
        print(
            colored(
                (
                    "Error in"
                    f" {self.__class__.__name__} pipeline:"
                    f" {error}"
```

```
        ),
        "red",
    )
)
```