Small Language Models

Jeff Prosise

@jprosise

Small Language Models (SLMs)

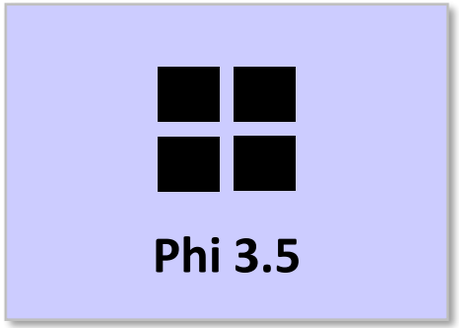
- Language models with fewer parameters for faster inference and smaller footprint but with most of the power of foundation models
  - Some run acceptably on CPUs and edge devices
  - Others perform very well on low-end GPUs (e.g., NVIDIA 3060 Ti)
- Models are open-source ("open weights") and can be run locally
  - Available from Hugging Face, Ollama, GitHub, and other sources
- Most are derived from flagship LLMs such as **Llama**, **Gemini Flash**, and **Mistral**
- Many are fine-tuned for specific tasks such as code generation

General-Purpose Models

- Scaled-down versions of foundation Large Language Models with many of the same capabilities

Llama 3.2

Distilled version of Meta's flagship **Llama 3.2** model featuring a **128K context window**. Available in **1B** and **3B** versions.



Phi 3.5

State-of-the-art multilingual model from Microsoft featuring **3.8B** parameters and a **128K context window**.

Gemma 2

Distilled version of **Gemini** that outperforms models twice its size. Features an **8K context window**. Available in **2B**, **7B**, and **27B** versions.

Mistral Small

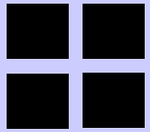
22B-parameter model from Mistral featuring a **128K context window**. Excels at NMT, document summarization, and other NLP tasks.

Code-Generation Models

- Lightweight models capable of generating code, performing code completion, and (in some cases) converting code to natural language

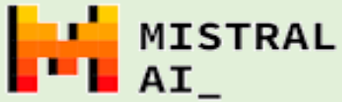
Code Llama 3.2

Distilled version of Meta's flagship Llama 3.2 model **fine-tuned for code generation**. Features a **128K context window**. Available in **1B** and **3B** versions.



Code Gemma

Distilled version of Gemini **fine-tuned for code generation**. Trained on Python, Java, JavaScript, C++, C#, Rust, Go, and more. Available in **2B** and **7B** versions.



Codestral

22B-parameter model with a **32K context window** that was trained on more than **80 languages**. Outperforms other code SLMs on many benchmarks.

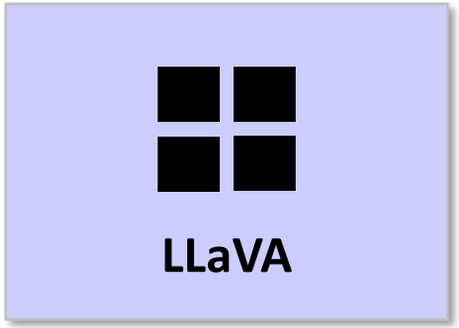
DeepSeek Coder 2

Code-generation model trained on more than 300 languages with a **128K context window**. Available in **16B** and **236B** versions.

Multimodal Models

- Multimodal LLMs that accept images as well as and rival much larger MLLMs in terms of image understanding

Multimodal model with **12B parameters** that surpasses larger models on some benchmarks. Supports multiple images with a **128K context window**.



Multimodal model that **rivals GPT-4** on benchmarks and was fine-tuned on the **Science QA dataset**. Available in **7B**, **13B**, and **34B** versions.

8B-parameter version of **LLaVA** built by fine-tuning Llama 3.

3.8B-parameter version of LLaVA built by **fine-tuning Phi-3 Mini**. Rivals LLaVA on benchmarks but runs faster.

Other Notable Models

- Dozens of models designed for specific tasks ranging from SQL code generation to question answering (RAG) to solving math problems
- Typically built by fine-tuning foundational SLMs

SQL Coder

SQL code-generation model created by **fine-tuning StarCoder** with 10,500+ curated samples. Available in **7B** and **15B** versions.

Qwen 2.5

Trained on **18 trillion tokens**. Available in seven versions ranging from **0.5B** parameters to **72B** with 128K context window.

Mathstral

7B-parameter model from Mistral designed for **math reasoning and scientific discovery** with 32K context window.

Nemotron Mini

4B-parameter model from NVIDIA optimized for **role play** and **retrieval-augmented generation** (RAG).

Ollama

- Open-source framework for running LLMs and SLMs locally
  - Versions available for Windows, macOS, and Linux
  - Automatically detects and uses GPU if present
- Invoke models by command line or API
  - SDKs available for Python and Node.js/JavaScript
- Provides easy access to dozens of open-source models, including **Llama 3.2** (1B and 3B versions), **Gemma 2** (2B/9B/27B), **CodeGemma** (2B/7B), Code Llama (7B/13B/34B/70B), **Mistral**, and **Phi-3**
  - See https://ollama.com/library for complete list

Generating Text with Llama 3.2

```
import ollama

messages = [{ 'role': 'user', 'content': 'Why is the sky blue?' }]

response = ollama.chat(
    model='llama3.2',
    messages=messages
)

print(response['message']['content'])
```

Streaming the Response

```
import ollama

messages = [{ 'role': 'user', 'content': 'Why is the sky blue?' }]

response = ollama.chat(
    model='llama3.2',
    messages=messages,
    stream=True
)

for chunk in response:
    print(chunk['message']['content'], end='')
```

Specifying the Temperature

```
import ollama

messages = [{ 'role': 'user', 'content': 'Generate a Python bubble sort function' }]

response = ollama.chat(
    model='llama3.2',
    messages=messages,
    options={ 'temperature': 0.2 },
    stream=True
)

for chunk in response:
    print(chunk['message']['content'], end='')
```

Generating JSON Output

```
prompt = '''
    Which books comprise the Harry Potter series? Please respond with a JSON array
    with individual books formatted as shown below. Do not output markdown.
    [
        {
            "title": "Harry Potter and the Prisoner of Azkaban",
            "year": 1999
        }
    ]
    '''

messages = [{ 'role': 'user', 'content': prompt }]
response = ollama.chat(model='llama3.2', messages=messages, format='json')
print(response['message']['content'])
```

Submitting an Image to LLaVA

```
messages = [{
    'role': 'user',
    'content': 'Describe what you see in this image',
    'images': ['IMAGE_PATH'] # Path to local image
}]

response = ollama.chat(
    model='llava:7b',
    messages=messages
)
```

Submitting Multiple Images to LLaVA

```
messages = [{
    'role': 'user',
    'content': 'Describe similarities between these images',
    'images': ['IMAGE_PATH_1', 'IMAGE_PATH_2'] # Paths to local images
}]

response = ollama.chat(
    model='llava:7b',
    messages=messages
)
```

Demo

Ollama

Hugging Face

- Hugging Face makes numerous SLMs available through its **transformers** package, including **Llama 3.2**, **Phi-3**, and **Gemma 2**
- Models can be run locally (in-process) without making API calls to a local server
- Some models require you to apply for access, receive an authentication token in return, and pass the token before loading the model

```
from huggingface_hub import login

# Authenticate to Hugging Face
login('HUGGING_FACE_TOKEN')
```

Generating Text with Gemma 2

```
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load the model and tokenizer
model_name = 'google/gemma-2-2b-it'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Tokenize the input
prompt = 'Why is the sky blue?'
tokens = tokenizer(prompt, return_tensors='pt')

# Generate the output
outputs = model.generate(**tokens, max_length=512)
print(tokenizer.decode(outputs[0]))
```

Streaming Text with Gemma 2

```python
from transformers import AutoTokenizer, AutoModelForCausalLM, TextStreamer

# Load the model and tokenizer
model_name = 'google/gemma-2-2b-it'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Tokenize the input
prompt = 'Why is the sky blue?'
tokens = tokenizer(prompt, return_tensors='pt')

# Stream the output
streamer = TextStreamer(tokenizer, skip_prompt=True) # Writes to stdout
_ = model.generate(**tokens, max_length=512, streamer=streamer)
```

Generating Text with Phi-3

```
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load the model and tokenizer
model_name = 'microsoft/Phi-3-mini-4k-instruct'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Tokenize the input
prompt = 'Why is the sky blue?'
tokens = tokenizer(prompt, return_tensors='pt')

# Generate the output
outputs = model.generate(**tokens, max_length=512)
print(tokenizer.decode(outputs[0]))
```

Submitting an Image to Pixtral

```
from vllm import LLM

messages = [{
    'role': 'user',
    'content': [
        { 'type': 'text', 'text': 'Describe what you see in this image' },
        { 'type': 'image_url', 'image_url': { 'url': 'IMAGE_URL' }} # Can be a data URL
    ]
}]

model = LLM(model='mistralai/Pixtral-12B-2409', tokenizer_mode='mistral')
response = model.chat(messages)
print(response[0].outputs[0].text)
```