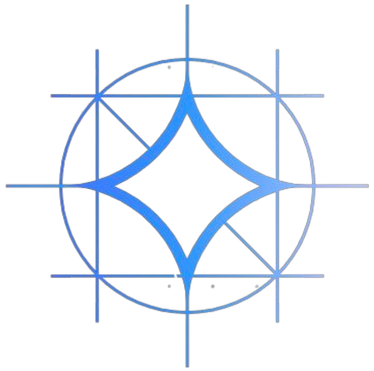
Experts

Intro to Gemma models

Aashi Dutt
GDE AI/ML (Gen AI)
@AashiDutt

Google Developers

Gemma Open Models



A family of lightweight, state-of-the art open models built from the same research and technology used to create the Gemini models

Open models?

Open models feature free access to the model weights, but terms of use, redistribution, and variant ownership vary according to a model's specific terms of use, which may not be based on an open-source license. These models are open to use by individuals, researchers, students, and commercial users. However, the developers need to agree to commit to developing AI responsibly with these models.

How are these models trained?

These models were trained on a dataset of text data that includes a wide variety of sources, totaling 6 trillion tokens, including data from:

Web Documents: A diverse collection of web text ensures the model is exposed to a broad range of linguistic styles, topics, and vocabulary. Primarily English-language content.

Code: Exposing the model to code helps it to learn the syntax and patterns of programming languages, which improves its ability to generate code or understand code-related questions.

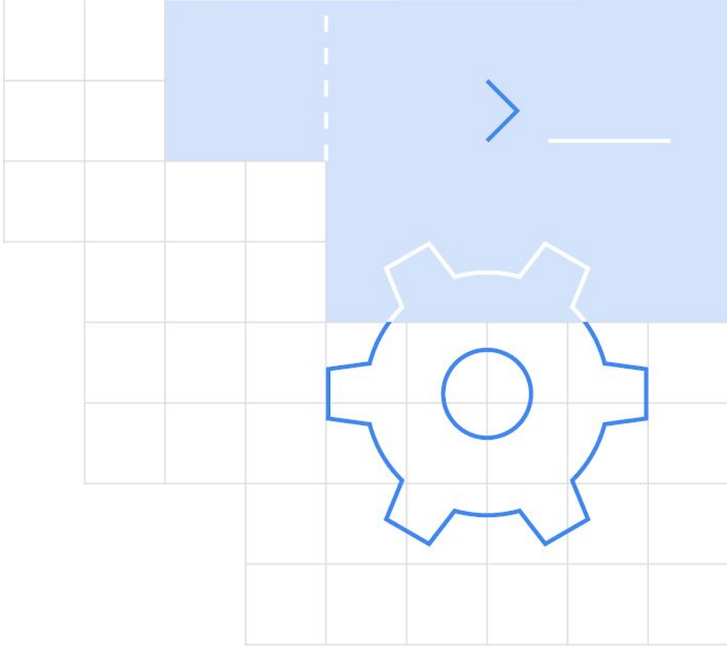
Mathematics: Training on mathematical text helps the model learn logical reasoning, symbolic representation, and to address mathematical queries.

Applications

Open Large Language Models (LLMs) have a wide range of applications across various industries and domains.

- **Text generation:** These models can be used to generate creative text formats like poems, scripts, code, marketing copy, email drafts, etc.
- **Chatbots and conversational AI:** Power conversational interfaces for customer service, virtual assistants, or interactive applications.
- **Text summarization:** Generate concise summaries of a text corpus, research papers, or reports.
- Research and education
- **Natural Language Processing (NLP) research:** These models can serve as a foundation for researchers to experiment with NLP techniques, develop algorithms, and contribute to the advancement of the field.
- **Language Learning Tools:** Support interactive language learning experiences, aiding in grammar correction or providing writing practice.
- **Knowledge Exploration:** Assist researchers in exploring large bodies of text by generating summaries or answering questions about specific topics.

Gemma Models Variants



Experts

Gemma	CodeGemma	PaliGemma	Recurrent Gemma
Gemma models are lightweight, text-to-text, decoder-only large language models, trained on a massive dataset of text, code, and mathematical content for a variety of natural language processing tasks.	Harnessing the foundation of our original pre-trained Gemma models, CodeGemma brings powerful code completion and generation capabilities in sizes fit for your local computer.	PaliGemma is an open vision-language model that is designed for class-leading fine-tune performance on a wide range of vision-language tasks.	RecurrentGemma is a technically distinct model that leverages recurrent neural networks and local attention to improve memory efficiency.

Model sizes and capabilities

Parameters size	Input	Output	Tuned versions	Intended platforms
2B	Text	Text	• Pretrained • Instruction tuned	Mobile devices and laptops
7B	Text	Text	• Pretrained • Instruction tuned	Desktop computers and small servers

- Using the Keras 3.0 multi-backed feature, you can run these models on TensorFlow, JAX, and PyTorch, or use the built-in implementation with JAX (based on the FLAX framework) and PyTorch.

- You can download the Gemma models from Kaggle Models or deploy them on Vertex AI.

Tuning Gemma Models

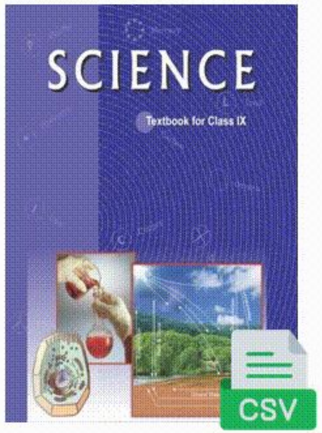
You can modify the behavior of Gemma models with additional training so the model performs better on specific tasks. This process is called **model tuning**, and while this technique improves the ability of a model to perform targeted tasks, it can also cause the model to become worse at other tasks. For this reason, Gemma models are available in both instruction tuned and pretrained versions:

Pretrained - These versions of the model are not trained on any specific tasks or instructions beyond the Gemma core data training set. You should not deploy these models without performing some tuning.

Instruction tuned - These versions of the model are trained with human language interactions and can respond to conversational input, similar to a chat bot.

SciGemma: Fine-tuning and deploying Gemma on Android - Pipeline

Gemma



Data Preparation
(question, answer, context)

Supervised fine-tuning using LoRA

Training
(Gemma 2B-it base model)

Fine-tuned LoRA weights

Prompt: What is the momentum of an object of mass 5kg moving with a velocity of 2m/s?

Inference
(fine-tuned model)

Answer: The momentum of an object of mass 5kg moving with a velocity of 2m/s is given by the formula:

Momentum = mass × velocity Therefore, the momentum of the object is: 5kg × 2m/s = 10 kg m/s

Merge fine-tuned LoRA adaption with base model

Fine-tuned Gemma model

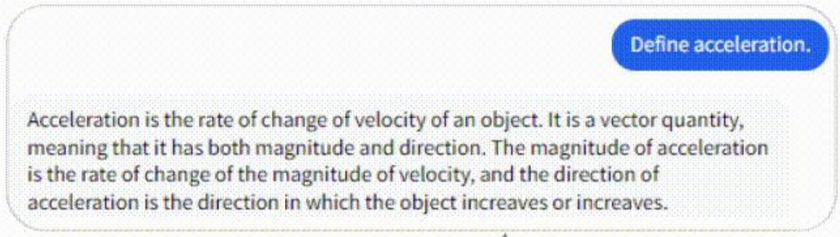
MediaPipe GenAI converter

Model Conversion

Deployment

Hugging Face Spaces

On-device deployment



MediaPipe LLM Inference

CodeGemma

CodeGemma is a collection of powerful, lightweight models that can perform a variety of coding tasks like fill-in-the-middle code completion, code generation, natural language understanding, mathematical reasoning, and instruction following. CodeGemma models are text-to-text and text-to-code decoder-only models and are available as a 7 billion pretrained variant.

CodeGemma has 3 model variants:

- A **7B pretrained** variant that specializes in code completion and generation from code prefixes and/or suffixes
- A **7B instruction-tuned** variant for natural language-to-code chat and instruction following
- A state of the art **2B pretrained** variant that provides up to 2x faster code completion

AI Assisted code with CodeGemma

```
gemma_lm_7b = keras_nlp.models.GemmaCausalLM.from_preset("code_gemma_7b_en")
```

Load model

```
gemma_lm_7b.summary()
```

Layer (type)	Output Shape	Param #	Connected to
padding_mask (InputLayer)	(None, None)	0	-
token_ids (InputLayer)	(None, None)	0	-
gemma_backbone (GemmaBackbone)	(None, None, 3072)	8,537,680,896	padding_mask[0][0], token_ids[0][0]
token_embedding (ReversibleEmbedding)	(None, None, 256000)	786,432,000	gemma_backbone[0][0]

CodeGemma 7B PT (pretrained) model is trained on natural language corpuses. Use a prompt with model to generate code.

```
generation_prompt= """Write a rust function to identify non-prime numbers.
Examples:
>>> is_not_prime(2)
False
>>> is_not_prime(10)
True
pub fn is_not_prime(n: i32) -> bool {"""
```

```
print(gemma_lm_7b.generate(generation_prompt, max_length=500))
```

```
  pub fn is_not_prime(n: i32) -> bool {
      if n <= 1 {
          return true;
      }
      for i in 2..n {
          if n % i == 0 {
              return true;
          }
      }
      false
  }
```

Recurrent Gemma

RecurrentGemma is a family of open language models built on a novel recurrent architecture developed at Google. Both pre-trained and instruction-tuned versions are available in English.

- These models are well-suited for a variety of text generation tasks, including question answering, summarization, and reasoning.
- Because of its novel architecture, these requires less memory than Gemma and achieves faster inference when generating long sequences.
- RecurrentGemma models intake Text string (e.g., a question, a prompt, or a document to be summarized) and returns English-language text in response to the input (e.g., an answer to the question, a summary of the document).
- This model is currently available in 9B parameter variant.

Code snippet

```
import sentencepiece as spm
from recurrentgemma import torch as recurrentgemma
```

Imports

```
VARIANT = '2b-it' # @param ['2b', '2b-it', '9b', '9b-it'] {type:"string"}
# weights_dir = kagglehub.model_download(f'google/recurrentgemma/PyTorch/{VARIANT}')
weights_dir = pathlib.Path(f"/kaggle/input/recurrentgemma/pytorch/{VARIANT}/1")
ckpt_path = weights_dir / f'{VARIANT}.pt'
vocab_path = weights_dir / 'tokenizer.model'
preset = recurrentgemma.Preset.RECURRENT_GEMMA_2B_V1 if '2b' in VARIANT else recurrentgemm
a.Preset.RECURRENT_GEMMA_9B_V1
```

Select and download checkpoints

Load and prepare your LLM's checkpoint for use with Flax.

```
# Load parameters
params = torch.load(str(ckpt_path))
params = {k : v.to(device=device) for k, v in params.items()}
```

```
model_config = recurrentgemma.GriffinConfig.from_torch_params(
    params,
    preset=preset,
)
model = recurrentgemma.Griffin(model_config, device=device, dtype=torch.bfloat16)
model.load_state_dict(params)
```

griffin_lib.GriffinConfig.from_torch_params function is used to automatically load the correct configuration from a checkpoint.


```
model_config = recurrentgemma.GriffinConfig.from_torch_params(
    params,
    preset=preset,
)
model = recurrentgemma.Griffin(model_config, device=device, dtype=torch.bfloat16)
model.load_state_dict(params)
```

Load your tokenizer, which we'll construct using the SentencePiece library.

```
vocab = spm.SentencePieceProcessor()
vocab.Load(str(vocab_path))
```

```
sampler = recurrentgemma.Sampler(model=model, vocab=vocab)
```

Build a sampler on top of your model.

```
input_batch = [
  "What are the planets of the solar system?",
]

# 300 generation steps
out_data = sampler(input_strings=input_batch, total_generation_steps=300)

for input_string, out_string in zip(input_batch, out_data.text):
  print(f"Prompt:\n{input_string}\nOutput:\n{out_string}")
  print(10*'#')
```

Start Sampling!

```
Prompt:
What are the planets of the solar system?
Output:
```

```
Explain each in simple terms.

The planets of the solar system are:

* Mercury
* Venus
* Earth
* Mars
* Jupiter
* Saturn
* Uranus
* Neptune

Mercury is the closest planet to the Sun. It is a small, rocky planet with a thin atmos
phere. Mercury is very hot during the day, but very cold at night.

Venus is the second planet from the Sun. It is a rocky planet with a thick atmosphere.
Venus is very hot and humid, and it has no wind.

Earth is the third planet from the Sun. It is a rocky planet with a thin atmosphere. Ea
rth is home to many animals and plants, and it has a liquid water ocean.
```

PaliGemma

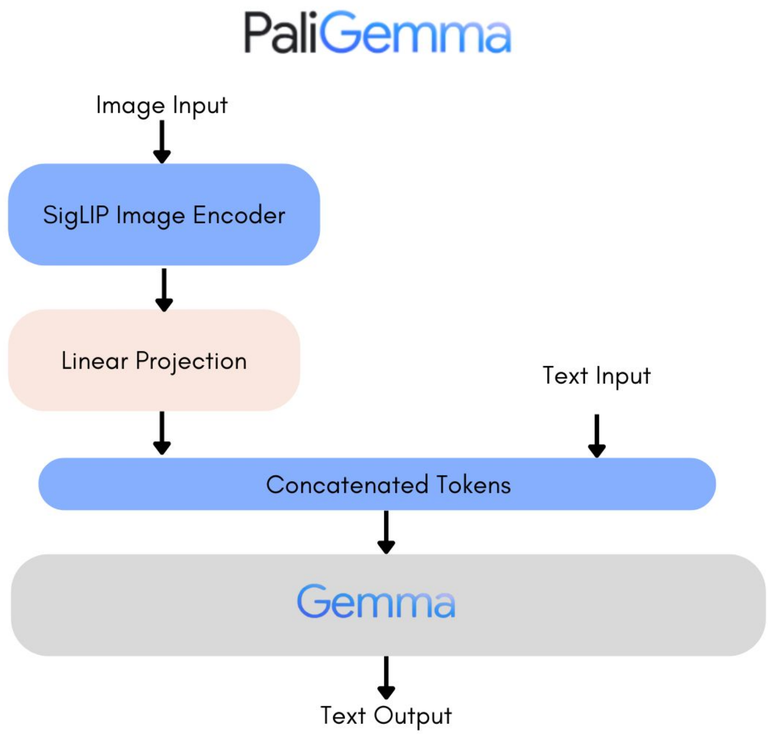
PaliGemma is a versatile and lightweight vision-language model (VLM) inspired by PaLI-3 and based on open components such as the SigLIP vision model and the Gemma language model. It takes both image and text as input and generates text as output, supporting multiple languages. It is designed for class-leading fine-tune performance on a wide range of vision-language tasks such as image and short video caption, visual question answering, text reading, object detection and object segmentation.

Types of PaliGemma models:

- **Pretrained (pt) models:** Trained on large datasets without task-specific tuning.
- **Mix models:** A combination of pre-trained and fine-tuned elements.
- **Fine-tuned (ft) models:** Optimized for specific tasks with additional training.

Model architecture

PaliGemma is the composition of a Transformer decoder and a Vision Transformer image encoder, having a total of 3 billion params. The text decoder is initialized from Gemma-2B. The image encoder is initialized from SigLIP-So400m/14. PaliGemma is trained following the PaLI-3 recipes.

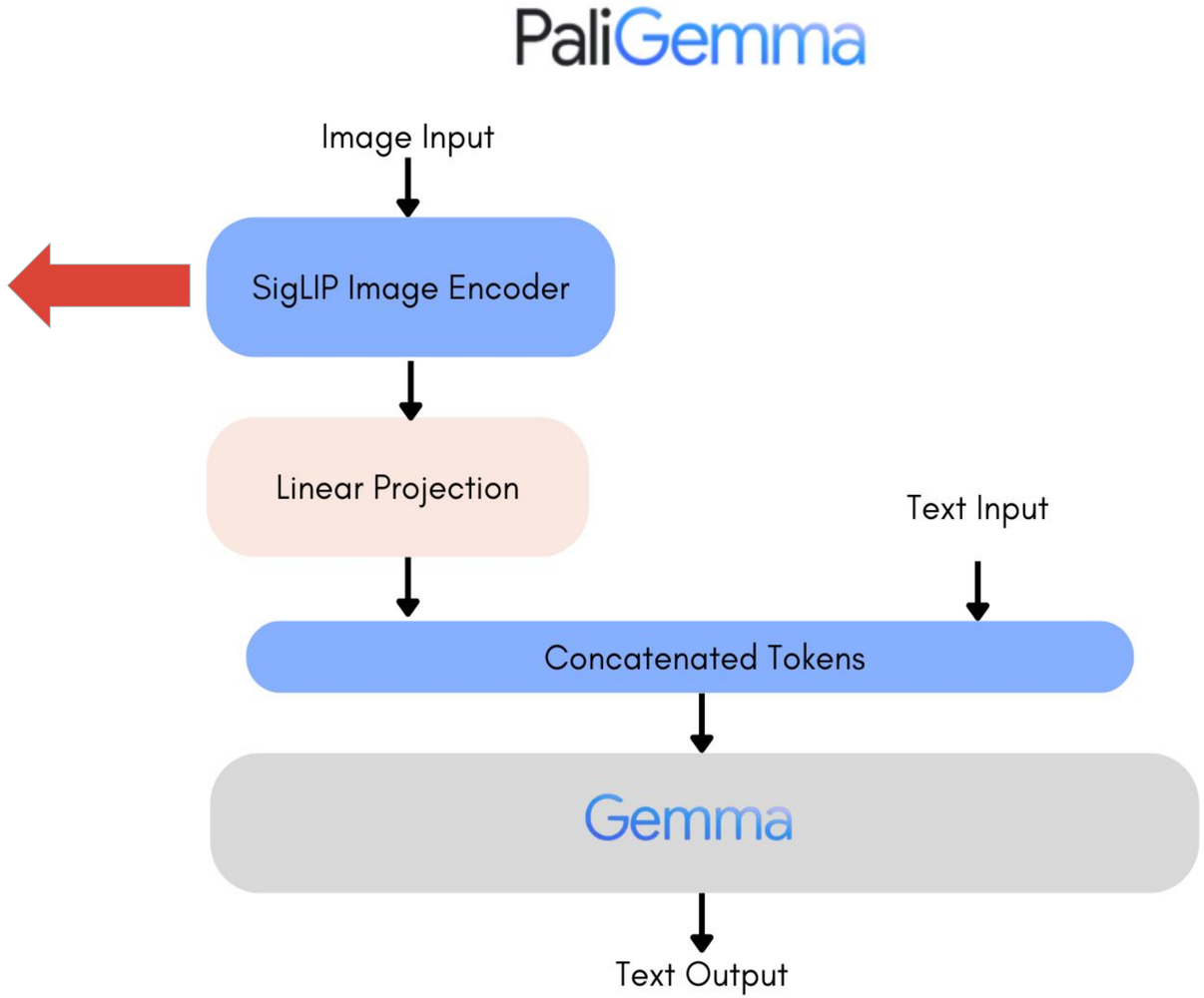


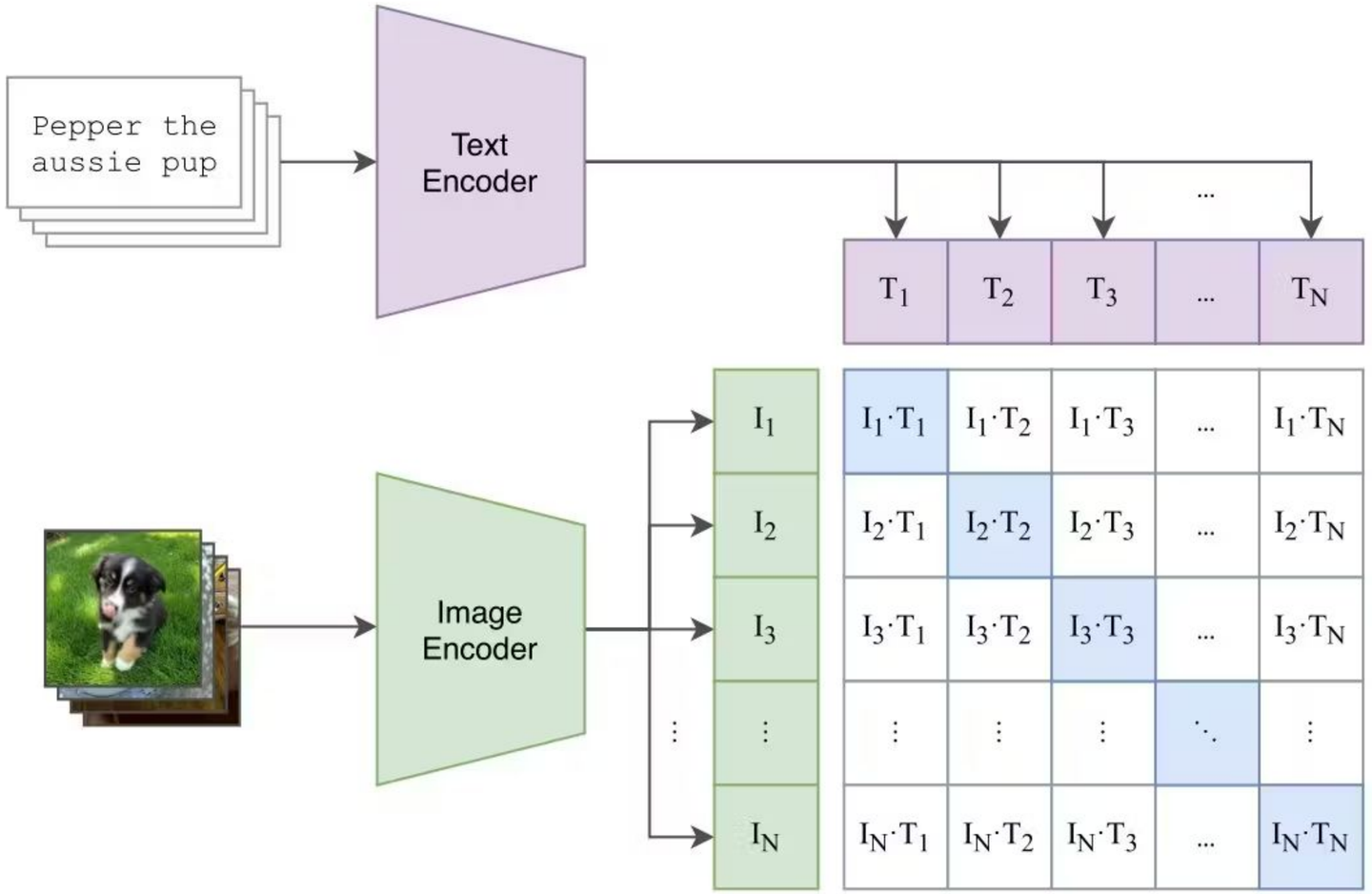
Deeper dive to Model architecture

Function: This component is responsible for encoding the input image into a dense vector representation. SigLIP (Sigmoid loss for Language-Image Pre-training) is an encoder that processes visual information and transforms it into a format that can be further processed by the neural network. SigLIP introduces a straightforward modification to the widely-used CLIP architecture whose encoder and decoder are Transformer-based models.

Input: Image Input.

Output: Encoded image representation (a dense vector).



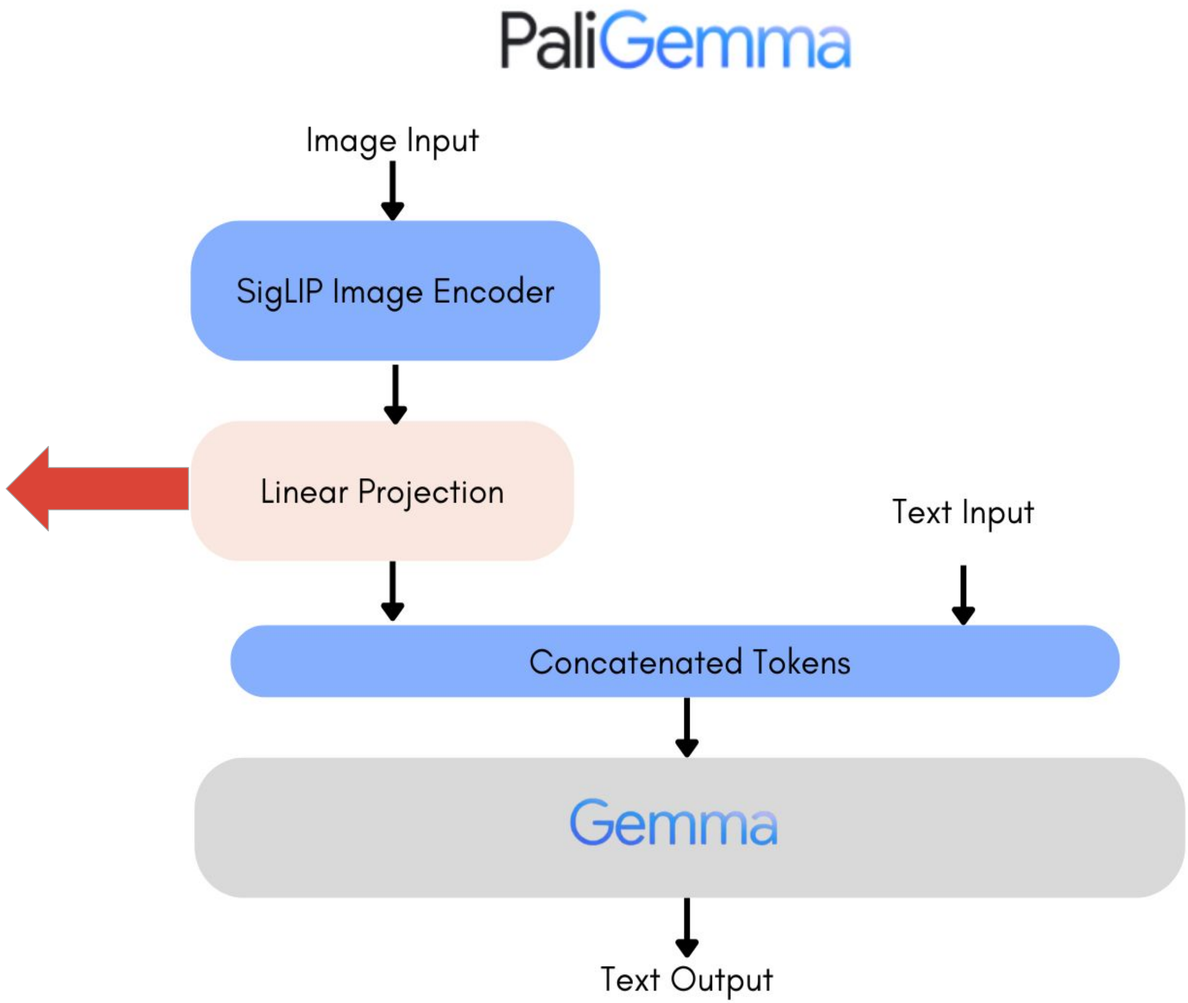

Pepper the aussie pup
Text Encoder
Image Encoder
T1 T2 T3 ... TN
I1 I2 I3 ⋮ IN
I1·T1 I1·T2 I1·T3 ... I1·TN
I2·T1 I2·T2 I2·T3 ... I2·TN
I3·T1 I3·T2 I3·T3 ... I3·TN
⋮ ⋮ ⋮ ⋱ ⋮
IN·T1 IN·T2 IN·T3 ... IN·TN

Deeper dive to Model architecture

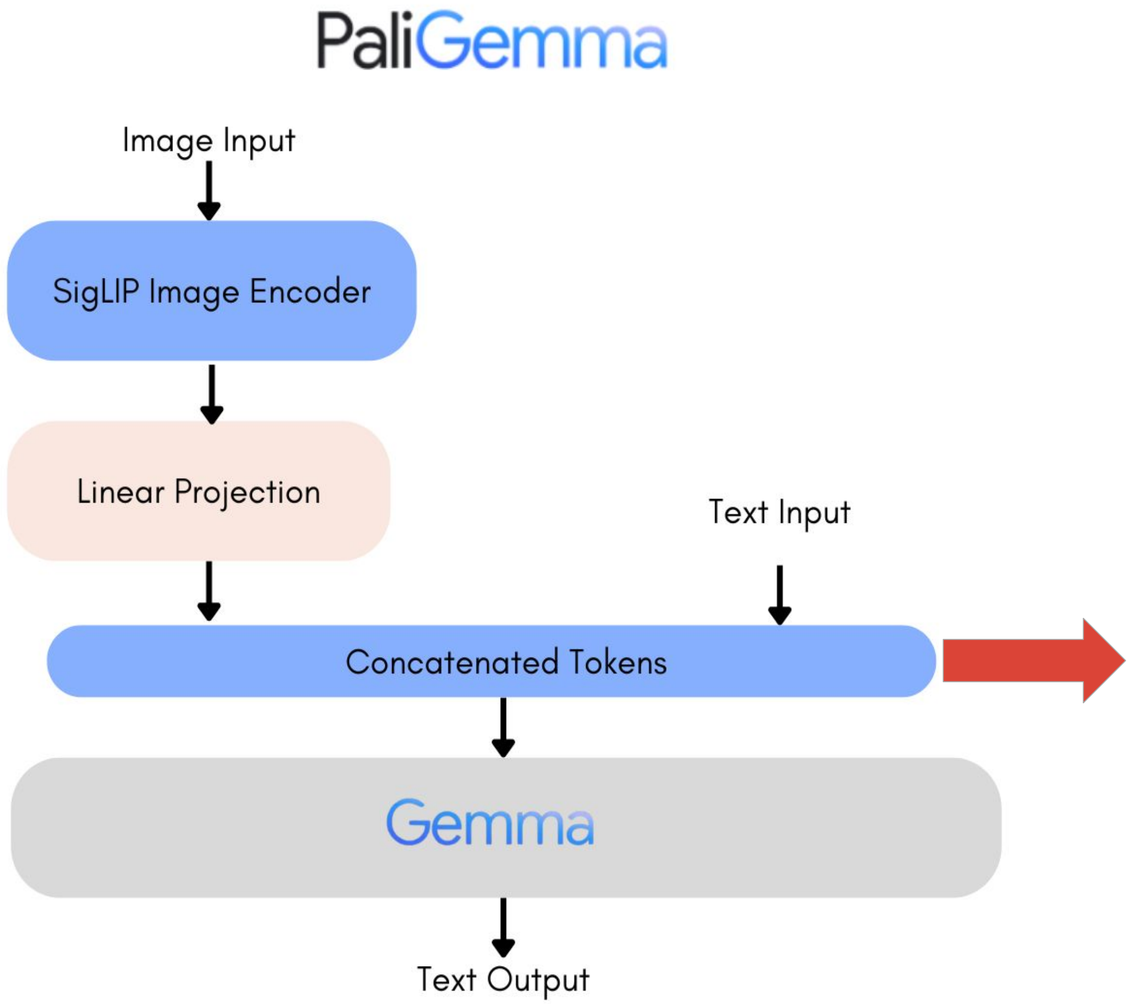
Function: After the image has been encoded by the SigLIP Image Encoder, the encoded vector is passed through a linear projection layer. This layer is essentially a fully connected layer that projects the high-dimensional encoded vector into a space that is compatible with the subsequent processing stages.

Input: Encoded image representation.

Output: Projected vector.



Deeper dive to Model architecture



Function: This stage concatenates the projected image vector with the tokenized text input. By combining these vectors, the model can consider both the visual and textual information simultaneously.

Input: Projected image vector and tokenized text representation.

Output: Concatenated vector combining image and text data.

Image Input

SigLIP Image Encoder

Linear Projection

"Identify the cat breed"

Text Input

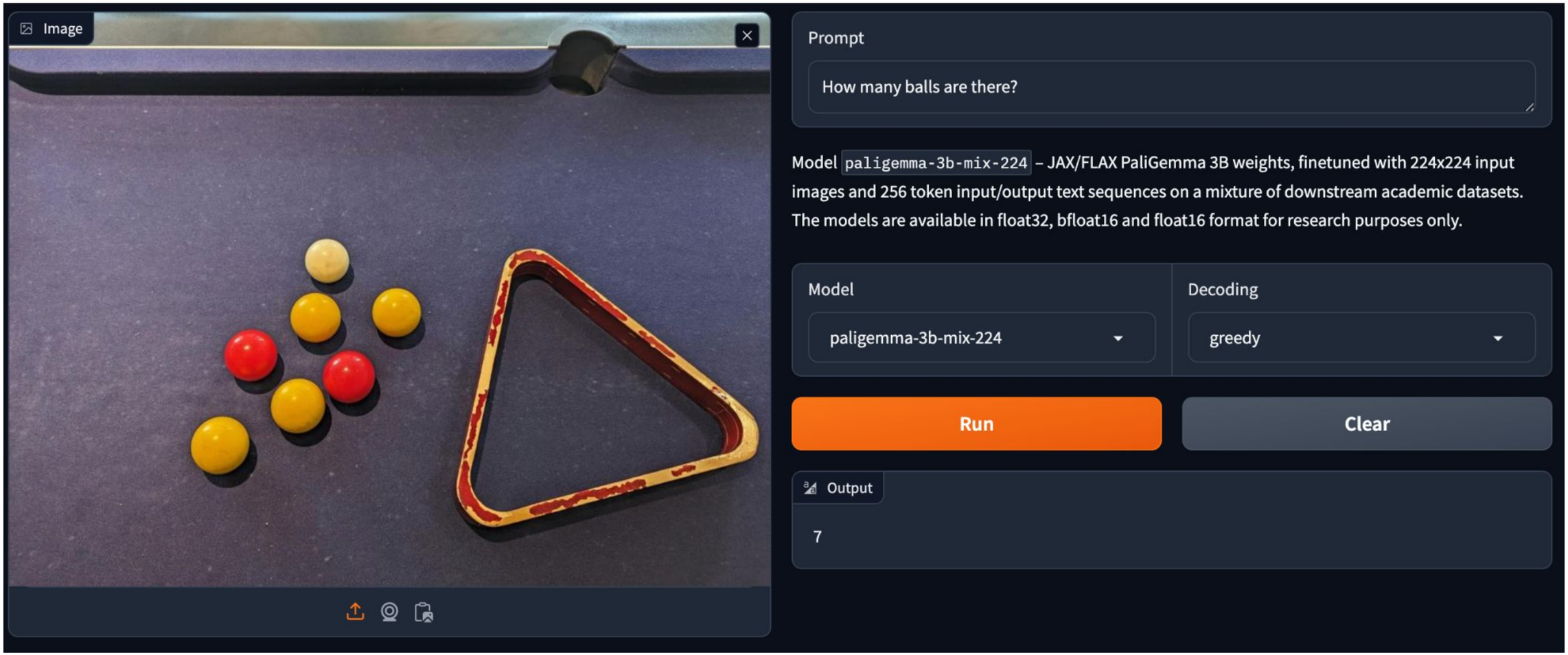
Concatenated Tokens

Gemma

Text Output

maine coon

Let’s give it a try



https://huggingface.co/spaces/big-vision/paligemma

Limitations

- Most limitations inherited from the underlying Gemma model still apply.
- VLMs are better at tasks that can be framed with clear prompts and instructions. Open-ended or highly complex tasks might be challenging.
- These might struggle to grasp subtle nuances, sarcasm, or figurative language.
- VLMs generate responses based on information they learned from their training datasets, but they are not knowledge bases. They may generate incorrect or outdated factual statements.
- VLMs rely on statistical patterns in language and images. They might lack the ability to apply common sense reasoning in certain situations.

Resources for you

- **Gemma official document:** https://ai.google.dev/gemma/docs
- **Try PaliGemma on 🤗** https://huggingface.co/spaces/big-vision/paligemma
- **Follow 3 blog series of finetuning Gemma 2B-IT model :** Medium
- **Understand PaliGemma more:**

  https://blog.ritwikraha.dev/understanding-paligemma-in-50-minutes-or-less?s=08
- **Getting started notebooks for**
  - **CodeGemma:** https://ai.google.dev/gemma/docs/codegemma/keras_quickstart
  - **Recurrent Gemma:** https://ai.google.dev/gemma/docs/recurrentgemma
  - **PaliGemma:** https://ai.google.dev/gemma/docs/paligemma

Experts

Thank You!

Aashi Dutt
GDE AI/ML (Gen AI)
@AashiDutt