Model Params

Valid Parameters and Values

Parameter	Description	Value Type	Example Usage
mirostat	Enable Mirostat sampling for controlling perplexity. (default: 0, 0 = disabled, 1 = Mirostat, 2 = Mirostat 2.0)	int	mirostat 0
mirostat_eta	Influences how quickly the algorithm responds to feedback from the generated text. A lower learning rate will result in slower adjustments, while a higher learning rate will make the algorithm more responsive. (Default: 0.1)	float	mirostat_eta 0.1
mirostat_tau	Controls the balance between coherence and diversity of the output. A lower value will result in more focused and coherent text. (Default: 5.0)	float	mirostat_tau 5.0
num_ctx	Sets the size of the context window used to generate the next token. (Default: 2048)	int	num_ctx 4096
repeat_last_n	Sets how far back for the model to look back to prevent repetition. (Default: 64, 0 = disabled, -1 = num_ctx)	int	repeat_last_n 64
repeat_penalty	Sets how strongly to penalize repetitions. A higher value (e.g., 1.5) will penalize repetitions more strongly, while a lower value (e.g., 0.9) will be more lenient. (Default: 1.1)	float	repeat_penalty 1.1
temperature	The temperature of the model. Increasing the temperature will make	float	temperature 0.7

Parameter	Description	Value Type	Example Usage
	the model answer more creatively. (Default: 0.8)		
seed	Sets the random number seed to use for generation. Setting this to a specific number will make the model generate the same text for the same prompt. (Default: 0)	int	seed 42
stop	Sets the stop sequences to use. When this pattern is encountered the LLM will stop generating text and return. Multiple stop patterns may be set by specifying multiple separate `stop` parameters in a modelfile.	string	stop "AI assistant:"
tfs_z	Tail free sampling is used to reduce the impact of less probable tokens from the output. A higher value (e.g., 2.0) will reduce the impact more, while a value of 1.0 disables this setting. (default: 1)	float	tfs_z 1
num_predict	Maximum number of tokens to predict when generating text. (Default: 128, -1 = infinite generation, -2 = fill context)	int	num_predict 42
top_k	Reduces the probability of generating nonsense. A higher value (e.g. 100) will give more diverse answers, while a lower value (e.g. 10) will be more conservative. (Default: 40)	int	top_k 40
top_p	Works together with top-k. A higher value (e.g., 0.95) will lead to more diverse text, while a lower value (e.g., 0.5) will generate more focused and conservative text. (Default: 0.9)		
num_batch	- Determines the number of tokens processed in parallel during inference. - Affects the speed of token	int	1024

Parameter	Description	Value Type	Example Usage
	generation and memory usage. - Default value is often 512, but can be adjusted to optimize performance. - Reducing num_batch can help solve Out of Memory (OOM) errors for large context models.		

Explanation

What's Top-p? 'Top-p' controls the diversity of AI responses: a low 'top-p' makes output more focused and predictable, while a high 'top-p' encourages variety and surprise.

Pair with temperature to fine-tune AI creativity:
higher temperatures with high 'top-p' for bold ideas, or
lower temperatures with low 'top-p' for precise answers.

Using top_p=1 essentially disables the "nucleus sampling" feature, where only the most probable tokens are considered.
This is equivalent to using full **softmax** probability distribution to sample the next word.

How Does Temperature Affect AI Outputs? Temperature controls the **randomness** of word selection. Lower temperatures lead to more predictable text, while higher temperatures allow for more novel text generation.

How Does Top-p Influence Temperature Settings in AI Language Models?
Top-p and temperature are both parameters that control the randomness of AI-generated text, but they influence outcomes in subtly different ways:

- **Low Temperatures (0.0 - 0.5):**
  - *Effect of Top-p:* A high `top_p` value will have minimal impact, as the model's output is already quite deterministic. A low `top_p` will further constrain the model, leading to very predictable outputs.

    - *Use Cases:* Ideal for tasks requiring precise, factual responses like technical explanations or legal advice. For example, explaining a scientific concept or drafting a formal business email.
- **Medium Temperatures (0.5 - 0.7):**
    - *Effect of Top-p:* `top_p` starts to influence the variety of the output. A higher `top_p` will introduce more diversity without sacrificing coherence.
    - *Use Cases:* Suitable for creative yet controlled content, such as writing an article on a current event or generating a business report that balances creativity with professionalism.
- **High Temperatures (0.8 - 1.0):**
    - *Effect of Top-p:* A high `top_p` is crucial for introducing creativity and surprise, but may result in less coherent outputs. A lower `top_p` can help maintain some coherence.
    - *Use Cases:* Good for brainstorming sessions, generating creative writing prompts, or coming up with out-of-the-box ideas where a mix of novelty and relevance is appreciated.
- **Extra-High Temperatures (1.1 - 2.0):**
    - *Effect of Top-p:* The output becomes more experimental and unpredictable, and `top_p`'s influence can vary widely. It's a balance between randomness and diversity.
    - *Use Cases:* Best for when you're seeking highly creative or abstract ideas, such as imagining a sci-fi scenario or coming up with a plot for a fantasy story, where coherence is less of a priority compared to novelty and uniqueness.

Certainly! I'll explain each of these parameters and their significance in language model inference:

```
# Good Results
PARAMETER temperature 0.6
PARAMETER top_k 80
PARAMETER top_p 0.8
PARAMETER frequency_penalty 0.9
PARAMETER num_ctx 16384
PARAMETER mirostat_eta 0.5
PARAMETER num_batch 1024
PARAMETER num_keep 256
```

```
PARAMETER num_thread 8
PARAMETER repeat_last_n 64
```

Temperature (0.6)

- Controls the randomness of the model's output.
- Range: 0.0 to 1.0 (and beyond, but rarely used above 1.0).
- At 0.6, it provides a balance between creativity and coherence.
- Lower values (closer to 0) make output more deterministic and focused.
- Higher values increase randomness and creativity.

Top K (60)

- Limits the number of top probable tokens considered for each generation step.
- Value 60 means only the 60 most likely next tokens are considered.
- Helps to reduce the chance of generating low-probability (and potentially nonsensical) tokens.

Top P (0.6)

- Also known as "nucleus sampling".
- Considers the smallest set of tokens whose cumulative probability exceeds the set value (0.6 in this case).
- Helps in maintaining a balance between diversity and quality of generated text.

Frequency Penalty (1.2)

- Reduces the likelihood of repeating the same words or phrases.
- Range: Usually 0.0 to 2.0.
- At 1.2, it applies a significant penalty to frequently used tokens, promoting diversity in the output.

Num_ctx (16384)

- Sets the context window size (in tokens).
- 16384 is quite large, allowing for processing of very long inputs or generating extended outputs.
- Larger context windows require more memory and processing power.

Mirostat_eta (0.5)

- Part of the Mirostat sampling algorithm, which dynamically adjusts the temperature.
- Eta controls the learning rate of the algorithm.
- 0.5 is a moderate value, allowing for adaptive sampling without being too aggressive.

Num_batch (1024)

- Determines how many tokens are processed in parallel during inference.
- 1024 is a relatively high value, which can speed up processing but requires more memory.

Num_keep (256)

- Specifies how many tokens from the start of the context to keep when truncating.
- Useful for maintaining important information when the context exceeds the maximum length.

Num_thread (8)

- Sets the number of CPU threads to use for computation.
- 8 threads suggest a multi-core processor, allowing for parallel processing.

Repeat_last_n (64)

- Related to the repeat penalty feature.
- Considers the last 64 tokens when applying the repetition penalty.
- Helps in reducing short-term repetitions in the generated text.

These parameters, when used together, aim to produce high-quality, diverse, and contextually relevant text while managing computational resources efficiently. The specific values chosen here suggest a balance between creativity and coherence, with a focus on handling long contexts and reducing repetition.

Yes, there are several other important parameters that affect the performance and behavior of large language models beyond those mentioned in the initial query. Here are some additional key parameters:

Learning Rate

- Controls how quickly the model updates its parameters during training.
- Affects the model's ability to converge on optimal solutions.

Dropout Rate

- Determines the probability of randomly "dropping out" neurons during training.
- Helps prevent overfitting by introducing regularization.

Attention Heads

- Specifies the number of parallel attention mechanisms in transformer-based models.
- More heads can capture different aspects of the input, potentially improving performance.

Hidden Layer Size

- Defines the dimensionality of the hidden layers in the neural network.
- Larger sizes can increase the model's capacity but also computational requirements.

Embedding Dimension

- Sets the size of the vector representations for tokens.
- Affects the model's ability to capture semantic relationships between words.

Warmup Steps

- Determines the number of training steps before the learning rate reaches its maximum value.
- Helps stabilize early training and can improve overall performance.

Sequence Length

- Specifies the maximum length of input sequences the model can process.
- Longer sequences allow for more context but increase computational demands.

Gradient Clipping

- Limits the maximum gradient value during training to prevent exploding gradients.
- Helps stabilize training, especially for very deep networks.

Seed

- Sets the initial random state for model initialization and training.
- Ensures reproducibility of results across different runs.

These parameters, along with those mentioned earlier (temperature, top_k, top_p, etc.), provide fine-grained control over various aspects of model training, inference, and generation. The optimal values for these parameters often depend on the specific task, dataset, and desired outcomes, requiring careful tuning and experimentation to achieve the best performance[1][2][3][4].

Citations:
[1] https://www.youtube.com/watch?v=PAbZRGGYNyM

[2] https://shecancode.io/blog/7-parameters-that-control-the-performance-of-large-language-models/
[3] https://kelvin.legal/understanding-large-language-models-what-are-paramters/
[4] https://www.linkedin.com/learning/introduction-to-large-language-models/what-are-parameters
[5] https://developers.google.com/machine-learning/resources/intro-llms

```
# Good Results
PARAMETER temperature 0.6
PARAMETER top_k 80
PARAMETER top_p 0.8
PARAMETER frequency_penalty 0.9
PARAMETER num_ctx 16384
PARAMETER mirostat_eta 0.5
PARAMETER num_batch 1024
PARAMETER num_keep 256
PARAMETER num_thread 8
PARAMETER repeat_last_n 64
```