With an **RTX 3060 (12GB VRAM)** and **64GB of system memory**, you can run moderately large language models, but you'll need to choose a model that fits within your hardware constraints. Here's a guide to help you select and run a DeepSeek model (or similar open-source models) on your system:

1. Model Size Considerations

VRAM (12GB): This limits the size of the model you can load into GPU memory.

Models with **up to 6-7 billion parameters** can typically fit in 12GB VRAM with optimizations (e.g., mixed precision, gradient checkpointing).

Larger models (e.g., 13B+ parameters) may require model parallelism or offloading to CPU/system memory, which can slow down inference.

System Memory (64GB): This allows you to load larger models into CPU memory if needed, but performance will be slower compared to GPU inference.

2. Recommended Model Sizes

Based on your hardware, here are some recommendations:

Small Models (1-3B Parameters)

These models will fit comfortably in your GPU memory and allow for fast inference.

Examples:

DeepSeek-V3-Small (if available)

GPT-2 (1.5B)

GPT-Neo (125M-2.7B)

LLaMA-7B (with 4-bit quantization)

Medium Models (6-7B Parameters)

These models can fit in 12GB VRAM with optimizations like **mixed precision** or **4-bit quantization**.

Examples:

DeepSeek-V3-Medium (if available)

LLaMA-7B (with quantization)

Falcon-7B

Mistral-7B

Large Models (13B+ Parameters)

These models will not fit entirely in 12GB VRAM but can be run with **CPU offloading** or **model parallelism**.

Examples:

LLaMA-13B (with 4-bit quantization)

Falcon-13B

DeepSeek-V3-Large (if available)

3. Optimizations for Running Models

To fit larger models into your hardware, use the following optimizations:

Quantization

Reduce the precision of the model weights (e.g., from 16-bit to 8-bit or 4-bit).

Libraries like **bitsandbytes** or **Hugging Face Transformers** support quantization.

Example:

python Copy

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

MODEL_NAME = "deepseek-ai/deepseek-v3-medium"  # Replace with your model
```

```
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForCausalLM.from_pretrained(MODEL_NAME, load_in_8bit=True, device_map="auto")
```

Mixed Precision
Use mixed precision (FP16) to reduce memory usage and speed up inference.
Example:

python Copy

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

MODEL_NAME = "deepseek-ai/deepseek-v3-medium"  # Replace with your model
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForCausalLM.from_pretrained(MODEL_NAME, torch_dtype=torch.float16).to("cuda")
```

CPU Offloading
Offload parts of the model to CPU memory when GPU memory is full.
Use libraries like **accelerate** or **DeepSpeed** for CPU offloading.
Example:

python Copy

```
from transformers import AutoModelForCausalLM, AutoTokenizer
from accelerate import init_empty_weights, load_checkpoint_and_dispatch

MODEL_NAME = "deepseek-ai/deepseek-v3-large"  # Replace with your model
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
with init_empty_weights():
    model = AutoModelForCausalLM.from_pretrained(MODEL_NAME)
model = load_checkpoint_and_dispatch(model, MODEL_NAME, device_map="auto")
```

Gradient Checkpointing
Reduce memory usage during training by trading compute for memory.
Example:

python Copy

```
model.gradient_checkpointing_enable()
```

4. Recommended Models for Your Hardware
Here are some specific models that should work well with your setup:
Small Models
GPT-2 (1.5B):
Easy to run, fits in GPU memory without optimizations.
Use case: General-purpose text generation.
GPT-Neo (2.7B):
Slightly larger than GPT-2, but still fits in 12GB VRAM.
Medium Models
LLaMA-7B (with 4-bit quantization):
Quantized version fits in 12GB VRAM.
Use case: High-quality text generation and reasoning.
Mistral-7B:
Optimized for performance and fits in 12GB VRAM with quantization.
Large Models

LLaMA-13B (with 4-bit quantization):

Requires CPU offloading but can run on your system.

Use case: Advanced tasks requiring larger context windows.

5. Example: Running a Quantized Model

Here's how you can run a quantized model using `bitsandbytes`:

python Copy

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

# Load a quantized model
MODEL_NAME = "deepseek-ai/deepseek-v3-medium"  # Replace with your model
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForCausalLM.from_pretrained(
    MODEL_NAME,
    load_in_8bit=True,  # Use 8-bit quantization
    device_map="auto"   # Automatically offload to CPU/GPU
)

# Process input data
input_text = "What is the capital of France?"
inputs = tokenizer(input_text, return_tensors="pt").to("cuda")
outputs = model.generate(**inputs, max_length=50)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

6. Tools to Monitor Resource Usage

Use `nvidia-smi` to monitor GPU memory usage.

Use `htop` or `task manager` to monitor CPU and system memory usage.

7. Next Steps

Start with a smaller model (e.g., GPT-2 or GPT-Neo) to test your setup.

Gradually move to larger models (e.g., LLaMA-7B or Mistral-7B) with quantization and optimizations.