# Upgrading DeckChatbot to LLaVA-LLaMA-3-8B: Integration Guide

## Introduction

This guide provides a comprehensive walkthrough for upgrading your DeckChatbot from the current 8GB version of LLaVA to the newer LLaVA-LLaMA-3-8B model from Hugging Face. Based on the examination of your project structure and current setup, we've developed specific recommendations to ensure a seamless transition while maximizing the benefits of the new model.

The LLaVA-LLaMA-3-8B model collection offers several variants with enhanced capabilities for image analysis and understanding, which aligns perfectly with your DeckChatbot's core functionality of interpreting blueprint images and extracting dimensions for deck design and quoting.

## 1. Model Selection and Integration

### Recommended Model Variant

From the available models in the collection [[1]](https://huggingface.co/collections/xtuner/llava-llama-3-8b-662a5f95adbe8d58799d7fdb), we recommend using the **llava-llama-3-8b-v1\_1-transformers** variant for the following reasons:

* It's the most downloaded/popular variant (419,563 downloads last month)
* It offers better performance metrics across benchmarks compared to other variants
* It supports the Transformers library interface, which provides flexibility for AWS integration
* It's fine-tuned with ShareGPT4V-PT and InternVL-SFT datasets, enhancing its visual understanding capabilities [[2]](https://huggingface.co/xtuner/llava-llama-3-8b-v1_1-transformers)

### Integration with AWS Setup

To integrate this model with your AWS setup, follow these steps:

1. **Setup Amazon SageMaker for Model Deployment**:
   * Create a SageMaker instance with sufficient resources (recommended: ml.g4dn.xlarge or larger)
   * Set up the Hugging Face Inference Toolkit on SageMaker [[20]](https://huggingface.co/docs/sagemaker/en/inference)
2. **Deploy the Model**:

* import sagemaker  
  from sagemaker.huggingface.model import HuggingFaceModel  
    
  # Initialize SageMaker session  
  sagemaker\_session = sagemaker.Session()  
    
  # Define model parameters  
  hub\_model\_id = "xtuner/llava-llama-3-8b-v1\_1-transformers"  
  role = "your-aws-iam-role"  
    
  # Create HuggingFace Model  
  huggingface\_model = HuggingFaceModel(  
   model\_data=None,  
   role=role,  
   transformers\_version="4.30.2",  
   pytorch\_version="2.0.1",  
   py\_version="py310",  
   hub={  
   'HF\_MODEL\_ID': hub\_model\_id,  
   'HF\_TASK': 'image-text-to-text',  
   }  
  )  
    
  # Deploy model to SageMaker endpoint  
  predictor = huggingface\_model.deploy(  
   initial\_instance\_count=1,  
   instance\_type="ml.g4dn.xlarge"  
  )

1. **Database Integration**:
   * Since your current setup uses SQLite for storing conversation history and measurements [[5]](https://github.com/alentwotime/deckchatbot-monorepo), no schema changes are needed
   * Update your database access layer to accommodate the new model's response format

## 2. Code Adjustments

### Backend API Changes

Update your backend API to interface with the new model. Based on your monorepo structure [[5]](https://github.com/alentwotime/deckchatbot-monorepo), these changes will likely be in the backend/backend-ai directory:

1. **Install required packages**:

* pip install transformers==4.30.2 torch accelerate

1. **Update Model Interface Code**:

* from transformers import pipeline, AutoProcessor, LlavaForConditionalGeneration  
  import torch  
  from PIL import Image  
  import requests  
    
  class LlavaModel:  
   def \_\_init\_\_(self, model\_id="xtuner/llava-llama-3-8b-v1\_1-transformers", device="cuda"):  
   self.model\_id = model\_id  
   self.device = device  
   self.processor = AutoProcessor.from\_pretrained(model\_id)  
   self.model = LlavaForConditionalGeneration.from\_pretrained(  
   model\_id,  
   torch\_dtype=torch.float16,  
   low\_cpu\_mem\_usage=True  
   ).to(device)  
     
   def generate\_response(self, image, prompt):  
   # Format the prompt according to LLaVA-3 expected format  
   formatted\_prompt = (  
   "<|start\_header\_id|>user<|end\_header\_id|>\n\n<image>\n"  
   f"{prompt}<|eot\_id|>"  
   "<|start\_header\_id|>assistant<|end\_header\_id|>\n\n"  
   )  
     
   # Process inputs  
   inputs = self.processor(formatted\_prompt, image, return\_tensors='pt').to(self.device, torch.float16)  
     
   # Generate response  
   output = self.model.generate(  
   \*\*inputs,   
   max\_new\_tokens=200,   
   do\_sample=False,  
   temperature=0.2,  
   top\_p=0.9  
   )  
     
   # Decode and return response  
   return self.processor.decode(output[0][2:], skip\_special\_tokens=True)

1. **AWS Integration**:

* import boto3  
  import json  
  import base64  
  from io import BytesIO  
    
  class SagemakerLlavaModel:  
   def \_\_init\_\_(self, endpoint\_name):  
   self.endpoint\_name = endpoint\_name  
   self.runtime = boto3.client('sagemaker-runtime')  
     
   def generate\_response(self, image, prompt):  
   # Format the prompt according to LLaVA-3 expected format  
   formatted\_prompt = (  
   "<|start\_header\_id|>user<|end\_header\_id|>\n\n<image>\n"  
   f"{prompt}<|eot\_id|>"  
   "<|start\_header\_id|>assistant<|end\_header\_id|>\n\n"  
   )  
     
   # Convert image to base64  
   buffer = BytesIO()  
   image.save(buffer, format="PNG")  
   image\_base64 = base64.b64encode(buffer.getvalue()).decode('utf-8')  
     
   # Prepare payload  
   payload = {  
   "inputs": {  
   "prompt": formatted\_prompt,  
   "image": image\_base64  
   },  
   "parameters": {  
   "max\_new\_tokens": 200,  
   "do\_sample": False,  
   "temperature": 0.2,  
   "top\_p": 0.9  
   }  
   }  
     
   # Invoke endpoint  
   response = self.runtime.invoke\_endpoint(  
   EndpointName=self.endpoint\_name,  
   ContentType='application/json',  
   Body=json.dumps(payload)  
   )  
     
   # Process response  
   result = json.loads(response['Body'].read().decode())  
   return result[0]['generated\_text']

### Frontend Changes

If your frontend directly communicates with the model, update the API calls to match the new format:

// Example frontend code update  
async function processDeckImage(image, question) {  
 const formData = new FormData();  
 formData.append('image', image);  
 formData.append('prompt', question);  
   
 const response = await fetch('/api/analyze-deck', {  
 method: 'POST',  
 body: formData  
 });  
   
 return response.json();  
}

## 3. Configuration Updates

### Updating from Ollama to Transformers Format

Your current setup uses Ollama with the following configuration [[6]](https://ollama.com/alentwotime/llava-deckbot:latest):

* Architecture: archllama
* Parameters: 8.03B
* Quantization: Q4\_K\_M
* Size: 4.9GB
* Temperature: 0.2
* Top\_p: 0.9

To update to the Transformers-based model, you'll need to:

1. **Update Environment Variables**:  
   Create or update .env file with:

* MODEL\_ID=xtuner/llava-llama-3-8b-v1\_1-transformers  
  USE\_SAGEMAKER=true  
  SAGEMAKER\_ENDPOINT=your-endpoint-name  
  AWS\_REGION=your-aws-region

1. **Update Docker Configuration**:  
   If you're using Docker, update your Dockerfile:

* FROM python:3.11-slim  
    
  WORKDIR /app  
    
  COPY requirements.txt .  
  RUN pip install --no-cache-dir -r requirements.txt  
    
  COPY . .  
    
  CMD ["python", "app.py"]

1. **Update requirements.txt**:

* transformers==4.30.2  
  torch>=2.0.1  
  accelerate>=0.20.3  
  pillow>=9.5.0  
  boto3>=1.28.0  
  fastapi>=0.100.0  
  uvicorn>=0.23.0  
  python-multipart>=0.0.6

1. **Create a Model Configuration File**:

* {  
   "model\_id": "xtuner/llava-llama-3-8b-v1\_1-transformers",  
   "parameters": {  
   "max\_new\_tokens": 200,  
   "temperature": 0.2,  
   "top\_p": 0.9  
   },  
   "prompt\_template": {  
   "system": "You are DeckChatbot AI. You interpret blueprint images, extract dimensions, and answer construction questions.",  
   "user\_prefix": "<|start\_header\_id|>user<|end\_header\_id|>\n\n<image>\n",  
   "user\_suffix": "<|eot\_id|>",  
   "assistant\_prefix": "<|start\_header\_id|>assistant<|end\_header\_id|>\n\n",  
   "assistant\_suffix": ""  
   }  
  }

## 4. Potential Challenges and Solutions

### Resource Requirements

**Challenge**: The new model may require more computational resources than your current setup.

**Solution**:

* Use GPU-accelerated instances on AWS (e.g., g4dn.xlarge or g5.xlarge)
* Implement efficient batching for multiple concurrent users
* Consider model quantization options for production deployment

### Performance Optimizations

**Challenge**: Response times may be slower with the larger model.

**Solution**:

* Implement request caching for common queries
* Use model quantization (FP16 instead of FP32)
* Optimize input resolutions (the model supports 336px)
* Consider parallel processing for image analysis tasks

### API Compatibility

**Challenge**: Different response formats between Ollama and Transformers-based models.

**Solution**:

* Create an adapter layer that normalizes responses
* Implement comprehensive testing to ensure compatibility
* Create fallback mechanisms in case of errors

### Cold Start Times

**Challenge**: SageMaker endpoints may have cold start delays.

**Solution**:

* Use SageMaker endpoint configurations with appropriate instance counts
* Implement warm-up strategies with periodic ping requests
* Consider using provisioned concurrency for consistent response times

## 5. Best Practices for Integration

### Incremental Deployment

* **Implement A/B Testing**: Run both models in parallel initially to compare performance
* **Gradual Rollout**: Start with a small percentage of users and gradually increase
* **Monitoring**: Set up comprehensive monitoring for model performance and errors

### Testing Strategy

* **Unit Tests**: Create comprehensive tests for the new model interface
* **Integration Tests**: Verify end-to-end flow with the new model
* **Performance Testing**: Benchmark response times and resource usage
* **Visual Regression Testing**: Ensure image analysis quality remains high

### Security Considerations

* **IAM Roles**: Create specific roles with least privilege for model access
* **Data Encryption**: Ensure all data in transit and at rest is encrypted
* **Input Validation**: Implement strict validation for all user inputs
* **Rate Limiting**: Set appropriate limits to prevent abuse

### Observability

* **Logging**: Implement structured logging for model interactions
* **Metrics**: Track latency, error rates, and throughput
* **Alerting**: Set up alerts for unusual behavior patterns
* **User Feedback**: Create mechanisms for collecting user feedback on model responses

## 6. Additional Recommendations

### Model Variants Considerations

Consider these alternatives depending on your specific needs:

1. **For reduced computational requirements**:
   * Use the GGUF quantized version: xtuner/llava-llama-3-8b-v1\_1-gguf [[11]](https://huggingface.co/xtuner/llava-llama-3-8b-v1_1-gguf)
   * This sacrifices some quality but reduces resource needs
2. **For maximum performance**:
   * Use the native format: xtuner/llava-llama-3-8b-v1\_1-hf [[10]](https://huggingface.co/xtuner/llava-llama-3-8b-hf)
   * This requires more resources but provides better quality

### AWS Architecture Improvements

1. **Implement Model Serving with AWS Lambda**:
   * Create a Lambda function for preprocessing requests
   * Use SageMaker for model inference
   * Implement API Gateway for request handling
2. **Optimize Cost with Amazon Elastic Inference**:
   * Attach accelerators to EC2 instances for cost-efficient inference
   * Scale based on demand patterns

### Feature Enhancements

Based on the capabilities of LLaVA-LLaMA-3-8B-v1\_1, consider implementing:

1. **Enhanced Blueprint Analysis**:
   * More accurate dimension extraction
   * Automatic identification of deck features
   * Material suggestion improvements
2. **Multi-image Context Support**:
   * Allow users to upload multiple images for context
   * Combine blueprint with site photos for better recommendations
3. **Interactive Guidance**:
   * More detailed step-by-step guidance for deck design
   * Real-time feedback on design modifications

## Conclusion

Upgrading to the LLaVA-LLaMA-3-8B model from Hugging Face represents a significant improvement for your DeckChatbot application. The new model offers enhanced image understanding capabilities, better performance on benchmark tests, and more accurate dimension extraction - all critical for your deck design and quoting tool.

By following this integration guide, you'll be able to leverage these advancements while ensuring a smooth transition from your current Ollama-based setup to a robust AWS-deployed solution. The configuration updates, code adjustments, and best practices outlined here will help you maintain performance while adding new capabilities to your application.

Remember to implement the integration incrementally, with thorough testing at each stage, to minimize disruption to your existing users while maximizing the benefits of this powerful new model.

1.[LLaVA-Llama-3-8B - a xtuner Collection](https://huggingface.co/collections/xtuner/llava-llama-3-8b-662a5f95adbe8d58799d7fdb)

2.[xtuner/llava-llama-3-8b-v1\_1-transformers](https://huggingface.co/xtuner/llava-llama-3-8b-v1_1-transformers)

3.[alentwotime/llava-deckbot](https://ollama.com/alentwotime/llava-deckbot)

4.[deckchatbot-monorepo/backend/backend-ai at main · alentwotime/deckchatbot-monorepo](https://github.com/alentwotime/deckchatbot-monorepo/tree/main/backend/backend-ai)

5.[GitHub - alentwotime/deckchatbot-monorepo](https://github.com/alentwotime/deckchatbot-monorepo)

6.[alentwotime/llava-deckbot](https://ollama.com/alentwotime/llava-deckbot:latest)

7.[deckchatbot-monorepo/Modelfile at main · alentwotime/deckchatbot-monorepo](https://github.com/alentwotime/deckchatbot-monorepo/blob/main/Modelfile)

8.[xtuner/llava-llama-3-8b](https://huggingface.co/xtuner/llava-llama-3-8b)

9.[xtuner/llava-llama-3-8b-transformers](https://huggingface.co/xtuner/llava-llama-3-8b-transformers)

10.[xtuner/llava-llama-3-8b-hf](https://huggingface.co/xtuner/llava-llama-3-8b-hf)

11.[xtuner/llava-llama-3-8b-v1\_1-gguf](https://huggingface.co/xtuner/llava-llama-3-8b-v1_1-gguf)

12.[Llava Llama-3 8B - a Hugging Face Space by MaziyarPanahi](https://huggingface.co/spaces/MaziyarPanahi/llava-llama-3-8b)

13.[README.md · xtuner/llava-llama-3-8b at ...](https://huggingface.co/xtuner/llava-llama-3-8b/blame/0e95fc83f63862e0094648d23624eacd0c50e9f0/README.md)

14.[llama3\_8b\_instruct\_clip\_vit\_larg...](https://github.com/InternLM/xtuner/issues/835)

15.[nsheth/llava-llama-3-8b-v1\_1-int4](https://ollama.com/nsheth/llava-llama-3-8b-v1_1-int4)

16.[A curated list of awesome Monorepo tools, software ...](https://github.com/korfuri/awesome-monorepo)

17.[Monorepo vs Multi-Repo — Through the Eyes of an AI Copilot](https://mihirdave95.medium.com/monorepo-vs-multi-repo-through-the-eyes-of-an-ai-copilot-c576d57eb41a)

18.[Make your monorepo feel small with Git's sparse index](https://github.blog/open-source/git/make-your-monorepo-feel-small-with-gits-sparse-index/)

19.[deploy a llava model on AWS SageMaker use ...](https://repost.aws/questions/QUEThjF6DVTr6uNM5KuSmNDg/deploy-a-llava-model-on-aws-sagemaker-use-huggingfacemodel)

20.[Deploy models to Amazon SageMaker](https://huggingface.co/docs/sagemaker/en/inference)

21.[Deploy LLaVA-1.5 on Amazon SageMaker | by Tom Gou](https://medium.com/@liltom.eth/deploy-llava-1-5-on-amazon-sagemaker-168b2efd2489)

22.[Create a Visual Chatbot on AWS EC2 with LLaVA-1.5](https://www.run.house/blog/create-a-visual-chatbot-on-aws-ec2-with-llava-1-5)

23.[How to perform inference with a Llava Llama model ...](https://stackoverflow.com/questions/77193088/how-to-perform-inference-with-a-llava-llama-model-deployed-to-sagemaker-from-hug)

24.[Deploy MoE-LLaVA Model in Amazon SageMaker](https://medium.com/@wang.fan420/deploy-moe-llava-model-in-amazon-sagemaker-68b803c529de)

25.[DEPLOY AND RUN HUGGING FACE MODELS IN AWS ...](https://www.youtube.com/watch?v=b4Qli13R_dA)

26.[Fine-tune large multimodal models using ...](https://aws.amazon.com/blogs/machine-learning/fine-tune-large-multimodal-models-using-amazon-sagemaker/)

27.[LLaVA-NeXT](https://huggingface.co/docs/transformers/main/en/model_doc/llava_next)

28.[How to Use Hugging Face Models with Ollama](https://danielmiessler.com/blog/how-to-use-hugging-face-models-with-ollama)

29.[Converting Hugging Face Models for Use with Ollama](https://medium.com/@udemirezen/converting-hugging-face-models-for-use-with-ollama-a-detailed-tutorial-4e64b66eea27)

30.[Use Ollama with any GGUF Model on Hugging Face Hub](https://huggingface.co/docs/hub/en/ollama)

31.[Exporting an Ollama Model to Hugging Face Format](https://www.reddit.com/r/ollama/comments/1iwp77w/exporting_an_ollama_model_to_hugging_face_format/)

32.[Run Any Hugging Face Model with Ollama in Just Minutes!](https://www.youtube.com/watch?v=jK_PZqeQ7BE)

33.[Ollama + HuggingFace - by Sudarshan Koirala](https://medium.com/@sudarshan-koirala/ollama-huggingface-8e8bc55ce572)

34.[How to convert HuggingFace model to GGUF format · ggml ...](https://github.com/ggml-org/llama.cpp/discussions/2948)

35.[How to run ANY Hugging Face Model in Ollama!!!!!!!!!!!!!!!!!!!!!!!!!!!!!](https://www.youtube.com/watch?v=r-_xykTmz_o)

36.[How to download a model and run it with Ollama locally?](https://discuss.huggingface.co/t/how-to-download-a-model-and-run-it-with-ollama-locally/77317)

37.[Use (Almost) Any Language Model Locally with Ollama ...](https://www.kdnuggets.com/use-almost-any-language-model-locally-with-ollama-and-hugging-face-hub)