Great! Here’s a **compact CNN model training script for MNIST**, followed by saving it as mnist\_cnn\_model.h5—perfect for Gradio + Hugging Face Spaces 🚀

## ✅ train\_mnist\_cnn.py – Train & Save CNN Model

import numpy as np

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.utils import to\_categorical

# 1. Load and preprocess data

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Reshape for CNN input (batch, height, width, channel)

x\_train = x\_train.reshape(-1, 28, 28, 1).astype("float32") / 255.0

x\_test = x\_test.reshape(-1, 28, 28, 1).astype("float32") / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# 2. Build CNN model

model = Sequential([

Conv2D(32, (3, 3), activation="relu", input\_shape=(28, 28, 1)),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(64, (3, 3), activation="relu"),

MaxPooling2D(pool\_size=(2, 2)),

Flatten(),

Dense(128, activation="relu"),

Dropout(0.3),

Dense(10, activation="softmax")

])

# 3. Compile & train

model.compile(optimizer="adam", loss="categorical\_crossentropy", metrics=["accuracy"])

model.fit(x\_train, y\_train, batch\_size=128, epochs=5, validation\_split=0.1)

# 4. Evaluate & save

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print(f"Test Accuracy: {test\_acc:.4f}")

# Save model for Gradio app

model.save("mnist\_cnn\_model.h5")

## 🔁 To Run Training:

python train\_mnist\_cnn.py

This will generate mnist\_cnn\_model.h5, which you can upload to your **Gradio Hugging Face Space** alongside app.py and requirements.txt.

Would you like me to:

Help you **upload this trained model and code to Hugging Face**?

Provide a **Colab notebook version** of the training for easy model generation?

Show how to **optimize this model (smaller/faster)** for deployment?

Let me know how you'd like to proceed!

Great choice! 🔥 Optimizing your CNN model for **faster inference** and **smaller file size** is essential for deploying on platforms like **Hugging Face Spaces**.

## ✅ Techniques to Optimize a CNN Model

### 1. ****Reduce Model Complexity****

Use fewer filters and layers

Reduce dense layer size

### ✅ Light Version of CNN for MNIST

Try this **lightweight architecture** (fast + small):

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to\_categorical

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train = x\_train.reshape(-1, 28, 28, 1).astype("float32") / 255.0

x\_test = x\_test.reshape(-1, 28, 28, 1).astype("float32") / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# Lightweight CNN model

model = Sequential([

Conv2D(16, (3, 3), activation="relu", input\_shape=(28, 28, 1)),

MaxPooling2D(pool\_size=(2, 2)),

Flatten(),

Dense(32, activation="relu"),

Dense(10, activation="softmax")

])

model.compile(optimizer="adam", loss="categorical\_crossentropy", metrics=["accuracy"])

model.fit(x\_train, y\_train, batch\_size=128, epochs=5, validation\_split=0.1)

model.evaluate(x\_test, y\_test)

model.save("mnist\_cnn\_light.h5")

✅ File size: ~150 KB  
🚀 Inference time: Very fast

### 2. ****Model Quantization (Optional)****

Use [TensorFlow Lite](https://www.tensorflow.org/lite) to quantize (reduce size up to **4x**) and accelerate inference.

import tensorflow as tf

# Load Keras model

model = tf.keras.models.load\_model("mnist\_cnn\_model.h5")

# Convert to TFLite with quantization

converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

tflite\_quant\_model = converter.convert()

# Save .tflite model

with open("mnist\_model\_quant.tflite", "wb") as f:

f.write(tflite\_quant\_model)

📌 Note: Gradio on Hugging Face doesn't support .tflite directly unless you use a custom TFLite interpreter.

## ✅ Summary: Best for Hugging Face

| **Method** | **Size** | **Speed** | **HF Compatibility** |
| --- | --- | --- | --- |
| Full CNN | ~1.2 MB | Fast | ✅ |
| Light CNN | ~150 KB | Super Fast | ✅ |
| Quantized TFLite | ~300 KB | Super Fast | ⚠️ (not direct) |

Would you like me to:

Upload a **pre-trained light CNN model** for you to use right away?

Help **benchmark prediction speed** in your Gradio app?

Or show how to **auto-convert model on training**?

Let me know how you'd like to level up next! 😄