Perfect 👍 since all your files are inside

D:\CattleBreedApp\ML\_Model

I’ll give you **ready-to-use code files** with correct paths and step-by-step instructions.

**✅ Folder structure (should look like this)**

D:\CattleBreedApp\ML\_Model

│

├─ Indian\_bovine\_breeds\ # your dataset (41 breed folders here) Kaggle dataset

├─ train\_model.py

├─ convert\_to\_tflite.py

├─ inspect\_tflite.py

├─ app\_streamlit.py

├─ requirements.txt

**📌 1. train\_model.py**

import os

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import numpy as np

# -------- Paths --------

dataset\_dir = r"D:\CattleBreedApp\ML\_Model\Indian\_bovine\_breeds"

model\_save\_path = r"D:\CattleBreedApp\ML\_Model\cattle\_breed\_model.h5"

labels\_out = r"D:\CattleBreedApp\ML\_Model\labels.txt"

img\_size = (224, 224)

batch\_size = 32

epochs = 8

# -------- Data generators --------

datagen = ImageDataGenerator(

rescale=1.0/255.0,

validation\_split=0.2,

rotation\_range=20,

width\_shift\_range=0.1,

height\_shift\_range=0.1,

horizontal\_flip=True,

zoom\_range=0.1

)

train\_gen = datagen.flow\_from\_directory(

dataset\_dir,

target\_size=img\_size,

batch\_size=batch\_size,

class\_mode="categorical",

subset="training",

shuffle=True

)

val\_gen = datagen.flow\_from\_directory(

dataset\_dir,

target\_size=img\_size,

batch\_size=batch\_size,

class\_mode="categorical",

subset="validation",

shuffle=False

)

num\_classes = train\_gen.num\_classes

print("Detected classes:", num\_classes)

# -------- Build model --------

base\_model = tf.keras.applications.MobileNetV2(

input\_shape=(\*img\_size, 3),

include\_top=False,

weights="imagenet"

)

base\_model.trainable = False

model = tf.keras.Sequential([

base\_model,

tf.keras.layers.GlobalAveragePooling2D(),

tf.keras.layers.Dense(num\_classes, activation="softmax")

])

model.compile(optimizer="adam",

loss="categorical\_crossentropy",

metrics=["accuracy"])

model.summary()

# -------- Train --------

history = model.fit(

train\_gen,

validation\_data=val\_gen,

epochs=epochs

)

# -------- Save model & labels --------

model.save(model\_save\_path)

print(f"✅ Saved Keras model to {model\_save\_path}")

index\_to\_name = {v: k for k, v in train\_gen.class\_indices.items()}

labels\_ordered = [index\_to\_name[i] for i in range(num\_classes)]

with open(labels\_out, "w", encoding="utf-8") as f:

f.write("\n".join(labels\_ordered))

print(f"✅ Saved labels.txt with {len(labels\_ordered)} breeds")

**📌 2. convert\_to\_tflite.py**

import tensorflow as tf

keras\_model = r"D:\CattleBreedApp\ML\_Model\cattle\_breed\_model.h5"

tflite\_model = r"D:\CattleBreedApp\ML\_Model\cattle\_breed\_model.tflite"

model = tf.keras.models.load\_model(keras\_model)

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

tflite\_data = converter.convert()

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

f.write(tflite\_data)

print(f"✅ Saved TFLite model to {tflite\_model}")

**📌 3. inspect\_tflite.py**

import tensorflow as tf

tflite\_model = r"D:\CattleBreedApp\ML\_Model\cattle\_breed\_model.tflite"

interpreter = tf.lite.Interpreter(model\_path=tflite\_model)

interpreter.allocate\_tensors()

input\_details = interpreter.get\_input\_details()

output\_details = interpreter.get\_output\_details()

print("Input details:", input\_details)

print("Output details:", output\_details)

➡️ Check that output\_details shows (1, 41) — this confirms your model has 41 outputs.

**📌 4. app\_streamlit.py**

import streamlit as st

import tensorflow as tf

import numpy as np

from PIL import Image

import os

MODEL\_PATH = r"D:\CattleBreedApp\ML\_Model\cattle\_breed\_model.tflite"

LABELS\_PATH = r"D:\CattleBreedApp\ML\_Model\labels.txt"

IMG\_SIZE = (224, 224)

TOP\_K = 5

# Load TFLite

interpreter = tf.lite.Interpreter(model\_path=MODEL\_PATH)

interpreter.allocate\_tensors()

input\_details = interpreter.get\_input\_details()[0]

output\_details = interpreter.get\_output\_details()[0]

# Load labels

with open(LABELS\_PATH, "r", encoding="utf-8") as f:

labels = [line.strip() for line in f.readlines() if line.strip()]

st.title("🐄 Indian Cattle Breed Classifier")

st.write("Upload an image to see top-5 predicted breeds")

uploaded\_file = st.file\_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

if uploaded\_file:

img = Image.open(uploaded\_file)

st.image(img, caption="Uploaded Image", width=400)

# Preprocess

img\_proc = img.convert("RGB").resize(IMG\_SIZE)

arr = np.array(img\_proc, dtype=np.float32) / 255.0

arr = np.expand\_dims(arr, axis=0)

interpreter.set\_tensor(input\_details["index"], arr.astype(input\_details["dtype"]))

interpreter.invoke()

preds = interpreter.get\_tensor(output\_details["index"])[0]

# Softmax normalize

exps = np.exp(preds - np.max(preds))

probs = exps / np.sum(exps)

if len(probs) != len(labels):

st.error(f"Model outputs {len(probs)} but labels.txt has {len(labels)} entries.")

else:

top\_idx = np.argsort(probs)[::-1][:TOP\_K]

st.success(f"✅ Best Prediction: {labels[top\_idx[0]]} ({probs[top\_idx[0]]\*100:.2f}%)")

st.subheader("📊 Top 5 Predictions")

for i in top\_idx:

st.write(f"\*\*{labels[i]}\*\* : {probs[i]\*100:.2f}%")

**📌 5. requirements.txt**

tensorflow>=2.11

streamlit

pillow

numpy

**🚀 Step-by-step process to run**

1. Open **Command Prompt / PowerShell** and go to your project:

cd D:\CattleBreedApp\ML\_Model

1. (Optional) Create venv:

python -m venv venv

venv\Scripts\activate

1. Install requirements:

pip install -r requirements.txt

1. Train the model (creates .h5 and labels.txt):

python train\_model.py

1. Convert to TFLite:

python convert\_to\_tflite.py

1. Inspect model output shape:

python inspect\_tflite.py

➡️ Make sure output shape = (1, 41).

1. Run the app:

streamlit run app\_streamlit.py

⚠️ If you still see **“output size (1) does not match labels count (41)”**, that means your model wasn’t trained correctly → final Dense layer must be Dense(num\_classes, activation="softmax").

👉 Do you want me to also give you a **fixed labels.txt file** directly (41 breeds listed), so you can test without retraining?