```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import image_dataset_from_directory
from sklearn.model_selection import KFold
from sklearn.utils import class_weight
from collections import Counter
import matplotlib.pyplot as plt

# Define dataset path
dataset_path = "/home/rgukt/Downloads/project/dataset_blood_group"
BATCH_SIZE = 32
IMAGE_SIZE = (64, 64)

# Load the dataset
dataset = image_dataset_from_directory(
    dataset_path,
    labels="inferred",
    label_mode="int",
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    shuffle=True
)

# Extract images and labels from the dataset
images = []
labels = []
for img, lbl in dataset.unbatch():
    images.append(img.numpy())
    labels.append(lbl.numpy())

images = np.array(images)
labels = np.array(labels)

# Define the number of folds
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Define the model architecture
def create_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(32, (3, 3), activation="relu",
padding="same", input_shape=(64, 64, 3)),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Dropout(0.3),

        tf.keras.layers.Conv2D(64, (3, 3), activation="relu",
padding="same"),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Dropout(0.4),

        tf.keras.layers.Conv2D(128, (3, 3), activation="relu",
padding="same"),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Dropout(0.4),

        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(256, activation="relu"),
        tf.keras.layers.Dropout(0.4),

        tf.keras.layers.Dense(len(dataset.class_names),
activation="softmax")
    ])

    model.compile(optimizer="adam",
                  loss="sparse_categorical_crossentropy",
                  metrics=["accuracy"])
    return model

# Perform k-fold cross-validation
fold_no = 1
accuracies = []
losses = []

for train_idx, val_idx in kf.split(images):
    print(f"Training on fold {fold_no}...")

    # Split data into training and validation sets
    train_images, train_labels = images[train_idx], labels[train_idx]
    val_images, val_labels = images[val_idx], labels[val_idx]

    # Create and train the model
    model = create_model()
    history = model.fit(
        train_images, train_labels,
        validation_data=(val_images, val_labels),
        epochs=20,  # Adjust the number of epochs
        batch_size=BATCH_SIZE,
        verbose=1,
    )

    # Evaluate the model on the validation set
    val_loss, val_accuracy = model.evaluate(val_images, val_labels,
verbose=0)
    accuracies.append(val_accuracy)
    losses.append(val_loss)

    print(f"Fold {fold_no} - Validation Accuracy: {val_accuracy},
Validation Loss: {val_loss}")
```

```python
    fold_no += 1

# Calculate average performance across all folds
print(f"Average Validation Accuracy: {np.mean(accuracies)}")
print(f"Average Validation Loss: {np.mean(losses)}")
```

```
Found 6000 files belonging to 8 classes.

2025-03-21 09:10:31.490514: I
tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is
aborting with status: OUT_OF_RANGE: End of sequence

Training on fold 1...

/home/rgukt/.local/lib/python3.10/site-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/20

2025-03-21 09:10:33.486382: W
external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 265420800 exceeds 10% of free system memory.

  1/169 ━━━━━━━━━━━━━━━━━━━━ 11:37 4s/step - accuracy: 0.0625 - loss:
224.5840

2025-03-21 09:10:38.615846: W
external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 24772608 exceeds 10% of free system memory.

  2/169 ━━━━━━━━━━━━━━━━━━━━ 54s 326ms/step - accuracy: 0.0938 - loss:
297.6440

2025-03-21 09:10:38.937690: W
external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 24772608 exceeds 10% of free system memory.

  3/169 ━━━━━━━━━━━━━━━━━━━━ 53s 320ms/step - accuracy: 0.1042 - loss:
328.7652

2025-03-21 09:10:39.239069: W
external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 24772608 exceeds 10% of free system memory.

  4/169 ━━━━━━━━━━━━━━━━━━━━ 52s 319ms/step - accuracy: 0.1074 - loss:
352.2149
```

```
2025-03-21 09:10:39.550238: W
external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 24772608 exceeds 10% of free system memory.

129/169 ━━━━━━━━━━━━━━━━━━━━ 10s 260ms/step - accuracy: 0.1293 - loss:
71.1938


# Save the best model (e.g., the one with the highest validation
accuracy)
best_model_index = np.argmax(accuracies)
best_model = create_model()
best_model.fit(images, labels, epochs=20, batch_size=BATCH_SIZE,
verbose=1)  # Train on the full dataset
best_model.save("/home/rgukt/Desktop/TF/best_model.h5")
print("Best model saved!")
```