```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Definir rutas de directorios de datos
train_data_dir = 'fruits/Training/'  # Directorio de entrenamiento
test_data_dir = 'fruits/Test/'  # Directorio de pruebas
```

En este bloque, se importa TensorFlow y las utilidades de generación de imágenes de Keras. Se definen las rutas de los directorios de datos de entrenamiento y pruebas, que contienen las imágenes de las frutas.

```
# Configurar generadores de imágenes
batch_size = 32
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2  # Dividir automáticamente los datos en conjuntos de entre
)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(100, 100),  # Redimensionar las imágenes a 100x100 píxeles (ajusta
    batch_size=batch_size,
    class_mode='categorical',  # Clasificación categórica
    subset='training'  # Utilizar el conjunto de entrenamiento
)

validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(100, 100),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'  # Utilizar el conjunto de validación
)
```

Se configura un generador de imágenes para el conjunto de entrenamiento. Las transformaciones de datos, como el escalado, el cambio de inclinación, el zoom y la inversión horizontal, se aplican a las imágenes para aumentar la variedad de datos. Además, el conjunto de datos se divide automáticamente en conjuntos de entrenamiento y validación, donde el 20% de los datos se utilizará para validación. Se crean generadores de datos de entrenamiento y validación utilizando las configuraciones de train_datagen. Estos generadores cargarán imágenes de los directorios de entrenamiento y aplicarán las transformaciones definidas. Además, se establecen opciones como el tamaño objetivo de las imágenes y la clasificación categórica.

```
# Crear un modelo de red neuronal convolucional
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')  # 4 clases de frutas
])
```

Se define un modelo de red neuronal convolucional (CNN) utilizando Keras. El modelo consta de capas de convolución, capas de pooling, una capa de aplanado (flatten), y capas densamente conectadas. El modelo se compone de una arquitectura simple para clasificar imágenes de frutas en 4 categorías.

```
# Compilar el modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy

# Entrenar el modelo
epochs = 10
model.fit(train_generator, validation_data=validation_generator, epochs=epochs)
```

Se compila el modelo utilizando el optimizador 'Adam', la función de pérdida 'categorical_crossentropy' (entrenamiento de clasificación categórica) y la métrica de 'accuracy' para evaluar el rendimiento del modelo durante el entrenamiento. Se inicia el proceso de entrenamiento del modelo. El modelo se entrena durante 10 épocas utilizando los datos generados por train_generator y se evalúa el rendimiento en el conjunto de validación proporcionado por validation_generator.

En resumen, este código es un ejemplo de entrenamiento de una red neuronal convolucional (CNN) para clasificar imágenes de frutas en 4 categorías utilizando TensorFlow y Keras. Se utiliza un generador de imágenes para cargar y preprocesar los datos, se crea el modelo de CNN, se compila el modelo y se realiza el entrenamiento. Este proceso es común en el aprendizaje profundo para tareas de clasificación de imágenes.