```
import tensorflow as tf
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout,Flatten,BatchNormalization,Conv2D,MaxPool2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import LearningRateScheduler

# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Data augmentation
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

# Preprocess and augment the training data
x_train = x_train / 255.0
x_test = x_test / 255.0
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Load the DenseNet-121 model with pretrained weights
#base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(32, 32, 3))

# Create a Sequential model
#model = Sequential()

# Add the DenseNet-121 model as the base
#model.add(base_model)

# Add custom classification head with dropout layers
#model.add(Flatten())
#model.add(Dense(1024, activation='relu'))
#model.add(Dropout(0.6))
#model.add(Dense(10, activation='softmax'))

# Freeze the layers in the base model
```

```
#for layer in base_model.layers:
   # layer.trainable = False

# Learning rate schedule
#def lr_schedule(epoch):
    #if epoch < 5:




INPUT_SHAPE = (32, 32, 3)
KERNEL_SIZE = (3, 3)
model = Sequential()

# Convolutional Layer
model.add(Conv2D(filters=32, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=32, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
# Pooling layer
model.add(MaxPool2D(pool_size=(2, 2)))
# Dropout layers
model.add(Dropout(0.5))

model.add(Conv2D(filters=64, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=64, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(filters=128, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(filters=256, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=KERNEL_SIZE, input_shape=INPUT_SHAPE, activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())
```

```
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))


def lr_schedule(epoch):
    if epoch < 15:
        return 0.001
    if epoch < 30:
        return 0.0001
    return 0.00001

# Compile the model
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model with data augmentation and learning rate scheduling
batch_size = 100
epochs = 30
model.fit(
    datagen.flow(x_train, y_train, batch_size=batch_size),
    steps_per_epoch=len(x_train) / batch_size,
    epochs=epochs,
    validation_data=(x_test, y_test),
    callbacks=[LearningRateScheduler(lr_schedule)])

# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Epoch 1/30
500/500 [==============================] - 44s 70ms/step - loss: 2.1388 - accuracy: 0.2087 - val_loss: 3.3427 - val_accuracy: 0.1723 - lr: 0.0010
Epoch 2/30
500/500 [==============================] - 34s 68ms/step - loss: 1.7798 - accuracy: 0.3287 - val_loss: 1.6972 - val_accuracy: 0.3792 - lr: 0.0010
Epoch 3/30
500/500 [==============================] - 33s 65ms/step - loss: 1.6212 - accuracy: 0.3974 - val_loss: 1.9390 - val_accuracy: 0.3728 - lr: 0.0010
Epoch 4/30
500/500 [==============================] - 34s 68ms/step - loss: 1.4873 - accuracy: 0.4691 - val_loss: 1.1949 - val_accuracy: 0.5749 - lr: 0.0010
Epoch 5/30
500/500 [==============================] - 34s 68ms/step - loss: 1.3494 - accuracy: 0.5271 - val_loss: 1.1167 - val_accuracy: 0.6047 - lr: 0.0010
Epoch 6/30
500/500 [==============================] - 33s 66ms/step - loss: 1.2467 - accuracy: 0.5677 - val_loss: 1.3599 - val_accuracy: 0.5589 - lr: 0.0010
Epoch 7/30
500/500 [==============================] - 34s 67ms/step - loss: 1.1697 - accuracy: 0.5968 - val_loss: 1.3417 - val_accuracy: 0.5512 - lr: 0.0010
Epoch 8/30
500/500 [==============================] - 33s 67ms/step - loss: 1.1089 - accuracy: 0.6212 - val_loss: 1.0025 - val_accuracy: 0.6700 - lr: 0.0010
Epoch 9/30
500/500 [==============================] - 35s 69ms/step - loss: 1.0373 - accuracy: 0.6473 - val_loss: 0.8123 - val_accuracy: 0.7206 - lr: 0.0010
Epoch 10/30
500/500 [==============================] - 34s 69ms/step - loss: 1.0018 - accuracy: 0.6628 - val_loss: 1.0257 - val_accuracy: 0.6703 - lr: 0.0010
Epoch 11/30
500/500 [==============================] - 33s 67ms/step - loss: 0.9628 - accuracy: 0.6751 - val_loss: 0.9655 - val_accuracy: 0.6787 - lr: 0.0010
Epoch 12/30
```

```
500/500 [==============================] - 33s 66ms/step - loss: 0.9311 - accuracy: 0.6856 - val_loss: 1.0088 - val_accuracy: 0.6458 - lr: 0.0010
Epoch 13/30
500/500 [==============================] - 33s 66ms/step - loss: 0.9060 - accuracy: 0.6942 - val_loss: 0.9353 - val_accuracy: 0.6942 - lr: 0.0010
Epoch 14/30
500/500 [==============================] - 34s 67ms/step - loss: 0.8829 - accuracy: 0.7046 - val_loss: 0.7802 - val_accuracy: 0.7382 - lr: 0.0010
Epoch 15/30
500/500 [==============================] - 34s 68ms/step - loss: 0.8604 - accuracy: 0.7112 - val_loss: 0.8947 - val_accuracy: 0.7077 - lr: 0.0010
Epoch 16/30
500/500 [==============================] - 33s 67ms/step - loss: 0.7921 - accuracy: 0.7329 - val_loss: 0.7086 - val_accuracy: 0.7633 - lr: 1.0000e-04
Epoch 17/30
500/500 [==============================] - 33s 66ms/step - loss: 0.7652 - accuracy: 0.7443 - val_loss: 0.6694 - val_accuracy: 0.7749 - lr: 1.0000e-04
Epoch 18/30
500/500 [==============================] - 34s 69ms/step - loss: 0.7513 - accuracy: 0.7491 - val_loss: 0.7026 - val_accuracy: 0.7653 - lr: 1.0000e-04
Epoch 19/30
500/500 [==============================] - 34s 67ms/step - loss: 0.7404 - accuracy: 0.7502 - val_loss: 0.6724 - val_accuracy: 0.7754 - lr: 1.0000e-04
Epoch 20/30
500/500 [==============================] - 33s 66ms/step - loss: 0.7351 - accuracy: 0.7525 - val_loss: 0.6603 - val_accuracy: 0.7821 - lr: 1.0000e-04
Epoch 21/30
500/500 [==============================] - 33s 66ms/step - loss: 0.7306 - accuracy: 0.7541 - val_loss: 0.7103 - val_accuracy: 0.7631 - lr: 1.0000e-04
Epoch 22/30
500/500 [==============================] - 33s 67ms/step - loss: 0.7340 - accuracy: 0.7563 - val_loss: 0.6443 - val_accuracy: 0.7842 - lr: 1.0000e-04
Epoch 23/30
500/500 [==============================] - 33s 67ms/step - loss: 0.7189 - accuracy: 0.7582 - val_loss: 0.6555 - val_accuracy: 0.7783 - lr: 1.0000e-04
Epoch 24/30
500/500 [==============================] - 33s 65ms/step - loss: 0.7161 - accuracy: 0.7616 - val_loss: 0.6239 - val_accuracy: 0.7902 - lr: 1.0000e-04
Epoch 25/30
500/500 [==============================] - 33s 66ms/step - loss: 0.7085 - accuracy: 0.7624 - val_loss: 0.6587 - val_accuracy: 0.7790 - lr: 1.0000e-04
Epoch 26/30
500/500 [==============================] - 33s 65ms/step - loss: 0.7135 - accuracy: 0.7611 - val_loss: 0.6521 - val_accuracy: 0.7804 - lr: 1.0000e-04
Epoch 27/30
500/500 [==============================] - 32s 65ms/step - loss: 0.7069 - accuracy: 0.7602 - val_loss: 0.6431 - val_accuracy: 0.7820 - lr: 1.0000e-04
Epoch 28/30
500/500 [==============================] - 33s 66ms/step - loss: 0.7051 - accuracy: 0.7614 - val_loss: 0.6429 - val_accuracy: 0.7806 - lr: 1.0000e-04
Epoch 29/30
```