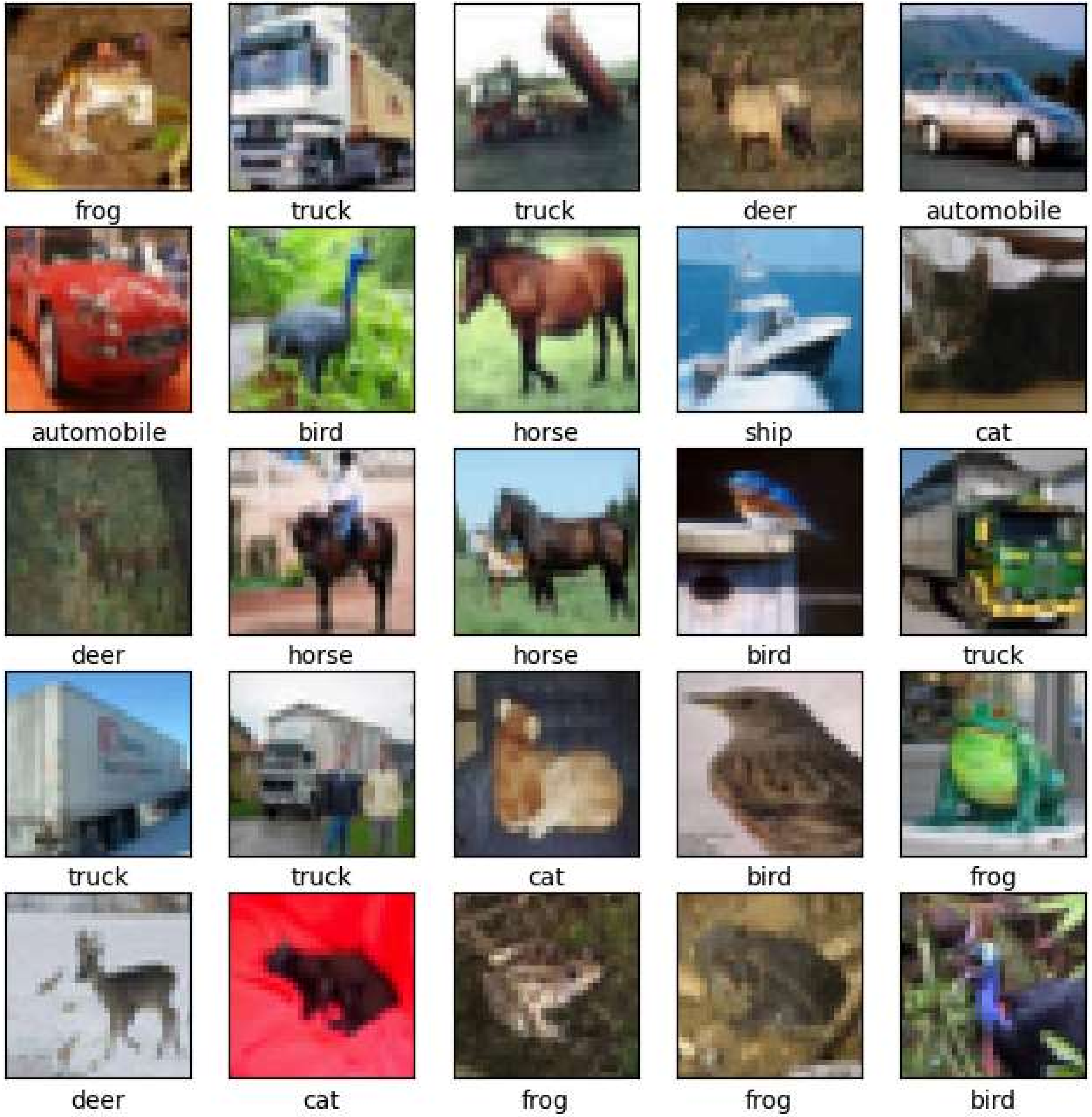
In [17]:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

# Load dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalize pixel values
x_train, x_test = x_train / 255.0, x_test / 255.0
```

In [18]:

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(8,8))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    # The CIFAR labels happen to be arrays,
    #which is why we need the extra index
    plt.xlabel(class_names[y_train[i][0]])
plt.show()
```



In [19]:

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

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
)
datagen.fit(x_train)
```

In [20]:

```
model = models.Sequential([
    layers.Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(32,32,3)),
    layers.BatchNormalization(),
    layers.Conv2D(32, (3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Dropout(0.25),

    layers.Conv2D(64, (3,3), padding='same', activation='relu'),
    layers.BatchNormalization(),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Dropout(0.35),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.5),
    layers.Dense(10, activation='softmax')
])
```

In [21]:

```
model.summary()
```

***Model: "sequential_1"***

| Layer (type) | Output Shape | Param # |
|---|---|---:|
| conv2d_4 (Conv2D) | (None, 32, 32, 32) | 896 |
| batch_normalization_6 (BatchNormalization) | (None, 32, 32, 32) | 128 |
| conv2d_5 (Conv2D) | (None, 30, 30, 32) | 9,248 |
| batch_normalization_7 (BatchNormalization) | (None, 30, 30, 32) | 128 |
| max_pooling2d_2 (MaxPooling2D) | (None, 15, 15, 32) | 0 |
| dropout_3 (Dropout) | (None, 15, 15, 32) | 0 |
| conv2d_6 (Conv2D) | (None, 15, 15, 64) | 18,496 |
| batch_normalization_8 (BatchNormalization) | (None, 15, 15, 64) | 256 |
| conv2d_7 (Conv2D) | (None, 13, 13, 64) | 36,928 |
| batch_normalization_9 (BatchNormalization) | (None, 13, 13, 64) | 256 |
| max_pooling2d_3 (MaxPooling2D) | (None, 6, 6, 64) | 0 |
| dropout_4 (Dropout) | (None, 6, 6, 64) | 0 |
| flatten_1 (Flatten) | (None, 2304) | 0 |
| dense_2 (Dense) | (None, 128) | 295,040 |
| batch_normalization_10 (BatchNormalization) | (None, 128) | 512 |
| dropout_5 (Dropout) | (None, 128) | 0 |
| dense_3 (Dense) | (None, 10) | 1,290 |

**Total params:** 363,178 (1.39 MB)

```
Trainable params: 362,538 (1.38 MB)
Non-trainable params: 640 (2.50 KB)
```

In [22]:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [23]:

```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

early_stop = EarlyStopping(patience=10, restore_best_weights=True)
lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=3)
```

In [24]:

```
layers.BatchNormalization()
```

Out[24]:

```
<BatchNormalization name=batch_normalization_11, built=False>
```

In [25]:

```
history = model.fit(datagen.flow(x_train, y_train, batch_size=64),
                    epochs=10,
                    validation_data=(x_test, y_test),
                    callbacks=[early_stop, lr_reduce])
```

```
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 126s 156ms/step - accuracy: 0.3293 - loss: 2.1206 - val_accuracy: 0.5225 - val_loss: 1.3
442 - learning_rate: 0.0010
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 113s 144ms/step - accuracy: 0.5225 - loss: 1.3333 - val_accuracy: 0.4927 - val_loss: 1.5
585 - learning_rate: 0.0010
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 109s 139ms/step - accuracy: 0.5847 - loss: 1.1737 - val_accuracy: 0.5722 - val_loss: 1.2
307 - learning_rate: 0.0010
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 111s 142ms/step - accuracy: 0.6242 - loss: 1.0630 - val_accuracy: 0.6754 - val_loss: 0.9
414 - learning_rate: 0.0010
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 110s 141ms/step - accuracy: 0.6529 - loss: 0.9857 - val_accuracy: 0.7186 - val_loss: 0.8
221 - learning_rate: 0.0010
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 111s 142ms/step - accuracy: 0.6655 - loss: 0.9510 - val_accuracy: 0.6932 - val_loss: 0.8
802 - learning_rate: 0.0010
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 110s 141ms/step - accuracy: 0.6860 - loss: 0.9046 - val_accuracy: 0.6888 - val_loss: 0.9
316 - learning_rate: 0.0010
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 112s 143ms/step - accuracy: 0.6924 - loss: 0.8895 - val_accuracy: 0.7153 - val_loss: 0.8
368 - learning_rate: 0.0010
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 111s 142ms/step - accuracy: 0.7140 - loss: 0.8232 - val_accuracy: 0.7435 - val_loss: 0.7
369 - learning_rate: 5.0000e-04
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 114s 146ms/step - accuracy: 0.7264 - loss: 0.7862 - val_accuracy: 0.7339 - val_loss: 0.7
860 - learning_rate: 5.0000e-04
```

In [26]:

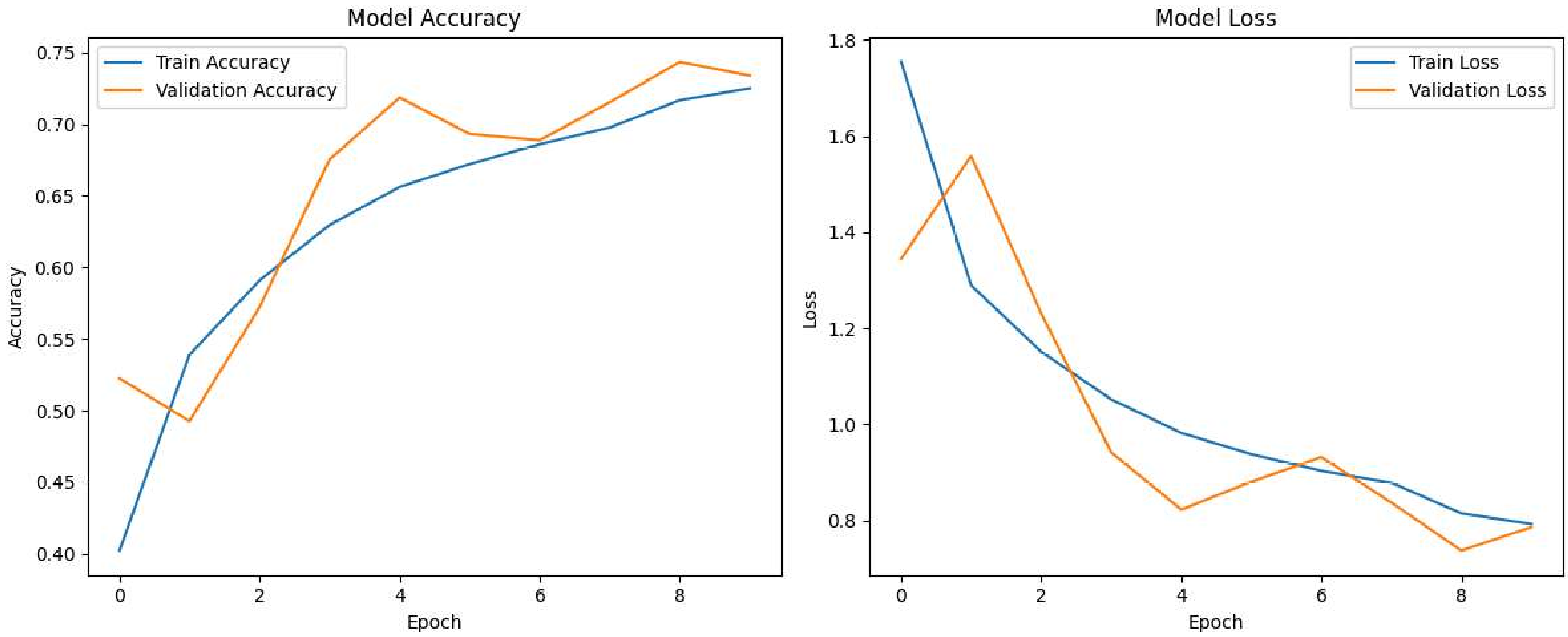
```python
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print("Test Accuracy:", test_acc)
```

```
313/313 - 4s - 12ms/step - accuracy: 0.7435 - loss: 0.7369
Test Accuracy: 0.7434999942779541
```

In [28]:

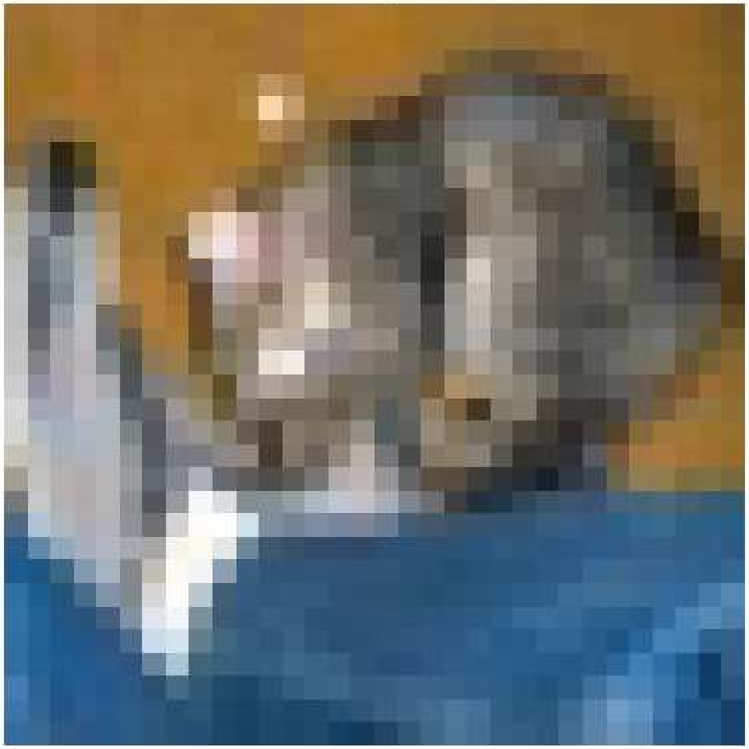
```python
import matplotlib.pyplot as plt

# Check if 'history' is defined
if 'history' not in locals():
        raise NameError("The variable 'history' is not defined. Please run the cell where model.fit() is called.")
```

```python
# Plot training & validation accuracy and loss values
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



```
In [30]: # Make predictions
         predictions = model.predict(x_test)
         # Display predictions for the first 5 test images
         for i in range(5):
             plt.imshow(x_test[i])
             plt.title(f"Predicted: {class_names[predictions[i].argmax()]}, Actual: {class_names[y_test[i][0]]}")
             plt.axis('off')
             plt.show()
         # Save the model
         model.save('cifar10_model.h5')
         # Load the model (if needed)
         # loaded_model = models.load_model('cifar10_model.h5')
         # loaded_model.summary()
         # Make predictions with the loaded model
```

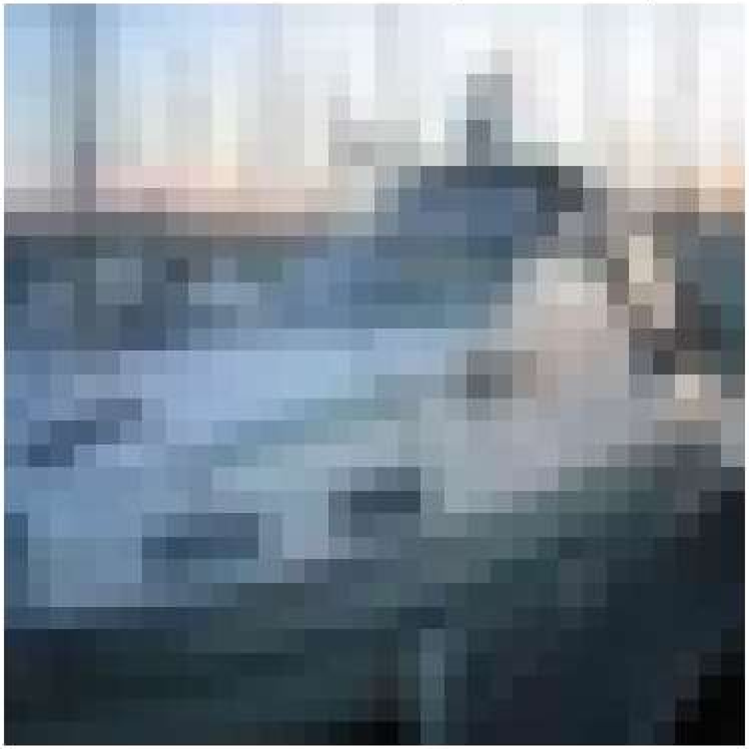
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step
```



Predicted: cat, Actual: cat

Predicted: ship, Actual: ship

Predicted: automobile, Actual: ship



Predicted: automobile, Actual: airplane

Predicted: frog, Actual: frog

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This f
ile format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.kera
s')` or `keras.saving.save_model(model, 'my_model.keras')`.
```