In [9]:
```python
import tensorflow as tf
from tensorflow.keras import layers, models
```

In [2]:
```python
# Define directory for train data
train_dir = "E:/Individual Projects/Deep Learning/archive/afhq/train"
```

In [3]:
```python
# Define directory for validation data
validation_dir = "E:/Individual Projects/Deep Learning/archive/afhq/val"
```

In [4]:
```python
# Define ImageDataGenerator for data augmentation and normalization
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,  # Normalize pixel values to [0, 1]
    rotation_range=40,  # Randomly rotate images by 40 degrees
    width_shift_range=0.2,  # Randomly shift images horizontally by 20%
    height_shift_range=0.2,  # Randomly shift images vertically by 20%
    shear_range=0.2,  # Shear intensity
    zoom_range=0.2,  # Randomly zoom into images by 20%
    horizontal_flip=True,  # Randomly flip images horizontally
    fill_mode='nearest'  # Fill mode for newly created pixels
)
```

In [5]:
```python
validation_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
```

In [6]:
```python
# Define batch size
batch_size = 32
```

In [7]:
```python
# Generate batches of augmented data from the directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),  # Resize images to 150x150
    batch_size=batch_size,
    class_mode='categorical'  # Use categorical labels
)

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(150, 150),
    batch_size=batch_size,
    class_mode='categorical'
)
```

```
Found 14630 images belonging to 3 classes.
Found 1500 images belonging to 3 classes.
```

In [10]:
```python
# Define the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(3, activation='softmax')  # Assuming 3 classes for your dataset
])
```

```
WARNING:tensorflow:From C:\Users\shash\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\shash\anaconda3\Lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [11]:
```python
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\shash\anaconda3\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer ins
tead.
```

In [12]:
```python
# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)
```

```
Epoch 1/10
WARNING:tensorflow:From C:\Users\shash\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTe
nsorValue instead.

WARNING:tensorflow:From C:\Users\shash\anaconda3\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compa
t.v1.executing_eagerly_outside_functions instead.

457/457 [==============================] - 728s 2s/step - loss: 0.8337 - accuracy: 0.5867 - val_loss: 0.4171 - val_accuracy: 0.8546
Epoch 2/10
457/457 [==============================] - 194s 424ms/step - loss: 0.4648 - accuracy: 0.8124 - val_loss: 0.3664 - val_accuracy: 0.8689
Epoch 3/10
457/457 [==============================] - 195s 427ms/step - loss: 0.3287 - accuracy: 0.8716 - val_loss: 0.2372 - val_accuracy: 0.9117
Epoch 4/10
457/457 [==============================] - 225s 492ms/step - loss: 0.2318 - accuracy: 0.9119 - val_loss: 0.1912 - val_accuracy: 0.9307
Epoch 5/10
457/457 [==============================] - 236s 516ms/step - loss: 0.1866 - accuracy: 0.9305 - val_loss: 0.1544 - val_accuracy: 0.9402
Epoch 6/10
457/457 [==============================] - 223s 488ms/step - loss: 0.1699 - accuracy: 0.9381 - val_loss: 0.1186 - val_accuracy: 0.9565
Epoch 7/10
457/457 [==============================] - 236s 517ms/step - loss: 0.1505 - accuracy: 0.9441 - val_loss: 0.0946 - val_accuracy: 0.9654
Epoch 8/10
457/457 [==============================] - 214s 469ms/step - loss: 0.1403 - accuracy: 0.9473 - val_loss: 0.1188 - val_accuracy: 0.9579
Epoch 9/10
457/457 [==============================] - 206s 451ms/step - loss: 0.1160 - accuracy: 0.9570 - val_loss: 0.1016 - val_accuracy: 0.9681
Epoch 10/10
457/457 [==============================] - 194s 425ms/step - loss: 0.1205 - accuracy: 0.9568 - val_loss: 0.0515 - val_accuracy: 0.9844
```

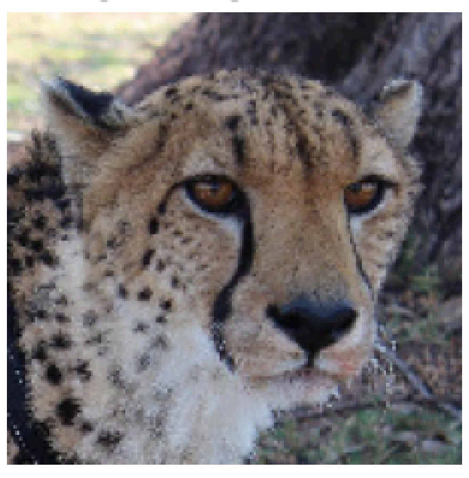
In [13]:
```python
# Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(validation_generator)
print('Test Loss:', test_loss)
print('Test Accuracy:', test_accuracy)
```

```
47/47 [==============================] - 9s 181ms/step - loss: 0.0512 - accuracy: 0.9840
Test Loss: 0.05120161548256874
Test Accuracy: 0.984000027179718
```

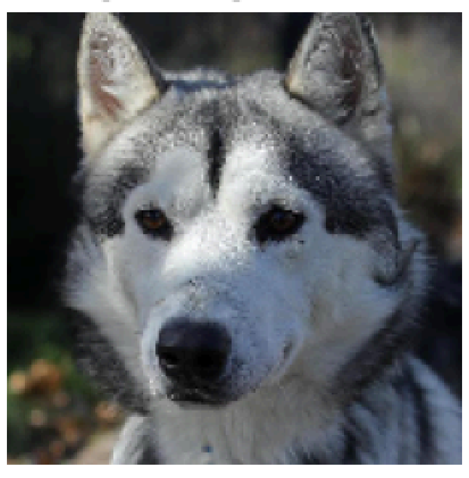
In [14]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Get a batch of test images and true labels
test_images_batch, true_labels_batch = next(validation_generator)

# Predict the labels for the test images
predicted_probabilities = model.predict(test_images_batch)
predicted_labels = np.argmax(predicted_probabilities, axis=1)

# Define a function to plot images with true and predicted labels
def plot_test_results(images, true_labels, predicted_labels):
    plt.figure(figsize=(10, 10))
    for i in range(min(len(images), 9)):  # Plot at most 9 images
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])
        plt.title(f"True: {true_labels[i]}, Predicted: {predicted_labels[i]}")
        plt.axis("off")
    plt.show()

# Plot the test results
plot_test_results(test_images_batch, true_labels_batch, predicted_labels)
```

```
1/1 [==============================] - 0s 200ms/step
```

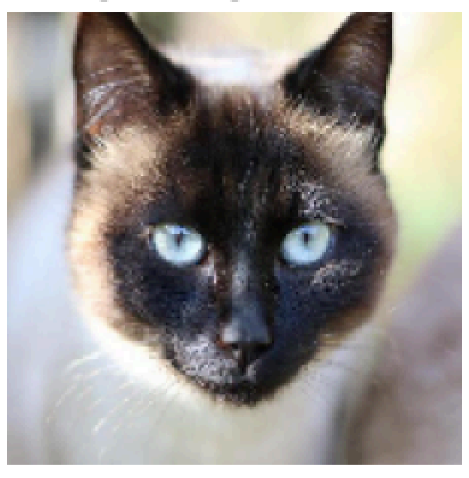
True: [0. 0. 1.], Predicted: 2



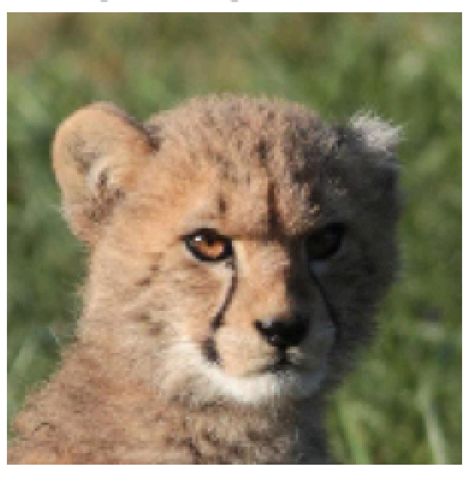
True: [0. 1. 0.], Predicted: 1



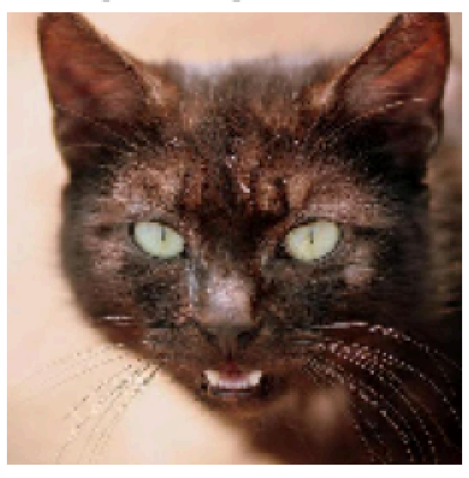
True: [1. 0. 0.], Predicted: 0



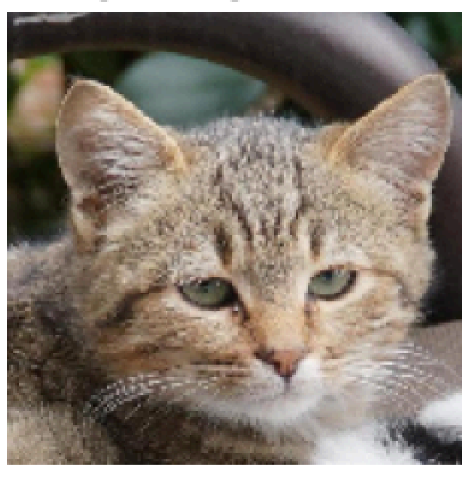
True: [0. 0. 1.], Predicted: 2



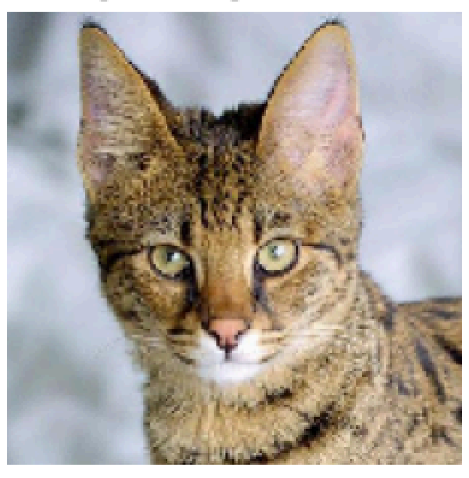
True: [1. 0. 0.], Predicted: 0



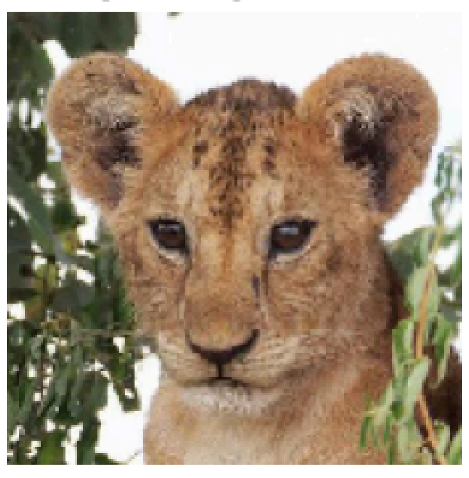
True: [1. 0. 0.], Predicted: 0



True: [1. 0. 0.], Predicted: 0



True: [0. 0. 1.], Predicted: 2



True: [1. 0. 0.], Predicted: 0