```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from PIL import Image
import warnings

DATADIR = '/PetImages'
CATEGORIES = ['Dog', 'Cat']

def create_data():
    data = []
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            img_path = os.path.join(path, img)
            try:
                with warnings.catch_warnings():
                    warnings.simplefilter("ignore")
                    with Image.open(img_path) as im:
                        im.verify()  # Verify if the image is not corrupt
                    img_array = cv2.imread(img_path, cv2.IMREAD_COLOR)
                    resized_array = cv2.resize(img_array, (150, 150))
                    data.append([resized_array, class_num])
            except Exception as e:
                pass
    return data

dataset = create_data()
np.random.shuffle(dataset)

X = []
y = []

for features, label in dataset:
    X.append(features)
    y.append(label)

X = np.array(X)
y = np.array(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)



import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import SparseCategoricalCrossentropy


# Normalize the pixel values
X_train_normalized = X_train / 255.0
X_test_normalized = X_test / 255.0

# Create a sequential model
model = Sequential([
    Flatten(input_shape=(150, 150, 3)),
    Dense(128, activation='relu'),
    Dense(2, activation='softmax')
])

# Compile the model
model.compile(optimizer=Adam(), loss=SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

# Train the model
model.fit(X_train_normalized, y_train, epochs=10, validation_split=0.1)

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(X_test_normalized, y_test)
print(f"Test accuracy: {test_accuracy}")
```

```
Epoch 1/10
/usr/local/lib/python3.9/dist-packages/keras/backend.py:5612: UserWarning: "`sparse_categorical_crossentropy` received `from_
```

```
  output, from_logits = _get_logits(
78/78 [==============================] - 17s 208ms/step - loss: 2.8103 - accuracy: 0.6967 - val_loss: 0.5617 - val_accuracy:
Epoch 2/10
78/78 [==============================] - 16s 201ms/step - loss: 0.6565 - accuracy: 0.7387 - val_loss: 0.5546 - val_accuracy:
Epoch 3/10
78/78 [==============================] - 16s 203ms/step - loss: 0.5403 - accuracy: 0.7706 - val_loss: 0.5696 - val_accuracy:
Epoch 4/10
78/78 [==============================] - 17s 216ms/step - loss: 0.5249 - accuracy: 0.7859 - val_loss: 0.5740 - val_accuracy:
Epoch 5/10
78/78 [==============================] - 16s 202ms/step - loss: 0.5421 - accuracy: 0.7557 - val_loss: 0.5474 - val_accuracy:
Epoch 6/10
78/78 [==============================] - 15s 197ms/step - loss: 0.4832 - accuracy: 0.7892 - val_loss: 0.5647 - val_accuracy:
Epoch 7/10
78/78 [==============================] - 15s 193ms/step - loss: 0.5315 - accuracy: 0.7718 - val_loss: 0.6296 - val_accuracy:
Epoch 8/10
78/78 [==============================] - 15s 194ms/step - loss: 0.4868 - accuracy: 0.7831 - val_loss: 0.5548 - val_accuracy:
Epoch 9/10
78/78 [==============================] - 16s 203ms/step - loss: 0.5008 - accuracy: 0.7872 - val_loss: 0.5574 - val_accuracy:
Epoch 10/10
78/78 [==============================] - 15s 194ms/step - loss: 0.4638 - accuracy: 0.7964 - val_loss: 0.5623 - val_accuracy:
22/22 [==============================] - 0s 20ms/step - loss: 0.5650 - accuracy: 0.7253
Test accuracy: 0.7252907156944275
```

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout

# Create a CNN model
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(2, activation='softmax')
])

# Compile the CNN model
cnn_model.compile(optimizer=Adam(), loss=SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

# Train the CNN model
cnn_model.fit(X_train_normalized, y_train, epochs=10, validation_split=0.1)

# Evaluate the CNN model on test data
test_loss, test_accuracy = cnn_model.evaluate(X_test_normalized, y_test)
print(f"Test accuracy: {test_accuracy}")
```

```
Epoch 1/10
78/78 [==============================] - 135s 2s/step - loss: 0.5927 - accuracy: 0.7702 - val_loss: 0.5799 - val_accuracy: 0.
Epoch 2/10
78/78 [==============================] - 131s 2s/step - loss: 0.5131 - accuracy: 0.7803 - val_loss: 0.5457 - val_accuracy: 0.
Epoch 3/10
78/78 [==============================] - 133s 2s/step - loss: 0.4902 - accuracy: 0.7831 - val_loss: 0.5231 - val_accuracy: 0.
Epoch 4/10
78/78 [==============================] - 120s 2s/step - loss: 0.4559 - accuracy: 0.7908 - val_loss: 0.5238 - val_accuracy: 0.
Epoch 5/10
78/78 [==============================] - 121s 2s/step - loss: 0.4130 - accuracy: 0.8106 - val_loss: 0.5842 - val_accuracy: 0.
Epoch 6/10
78/78 [==============================] - 122s 2s/step - loss: 0.3651 - accuracy: 0.8380 - val_loss: 0.5373 - val_accuracy: 0.
Epoch 7/10
78/78 [==============================] - 121s 2s/step - loss: 0.3083 - accuracy: 0.8582 - val_loss: 0.5998 - val_accuracy: 0.
Epoch 8/10
78/78 [==============================] - 124s 2s/step - loss: 0.2512 - accuracy: 0.8946 - val_loss: 0.6769 - val_accuracy: 0.
Epoch 9/10
78/78 [==============================] - 118s 2s/step - loss: 0.1868 - accuracy: 0.9249 - val_loss: 0.7257 - val_accuracy: 0.
Epoch 10/10
78/78 [==============================] - 121s 2s/step - loss: 0.1241 - accuracy: 0.9548 - val_loss: 0.9244 - val_accuracy: 0.
22/22 [==============================] - 8s 385ms/step - loss: 0.7226 - accuracy: 0.7558
Test accuracy: 0.7558139562606812
```

```
# Create a CNN model
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
```

```
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(2, activation='softmax')
])

# Compile the CNN model
cnn_model.compile(optimizer=Adam(), loss=SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

# Train the CNN model
cnn_model.fit(X_train_normalized, y_train, epochs=10, validation_split=0.1)

# Evaluate the CNN model on test data
test_loss, test_accuracy = cnn_model.evaluate(X_test_normalized, y_test)
print(f"Test accuracy: {test_accuracy}")
```

```
Epoch 1/10
78/78 [==============================] - 132s 2s/step - loss: 0.5318 - accuracy: 0.7738 - val_loss: 0.5432 - val_accuracy: 0.
Epoch 2/10
78/78 [==============================] - 121s 2s/step - loss: 0.4769 - accuracy: 0.7807 - val_loss: 0.5429 - val_accuracy: 0.
Epoch 3/10
78/78 [==============================] - 118s 2s/step - loss: 0.4531 - accuracy: 0.7985 - val_loss: 0.5414 - val_accuracy: 0.
Epoch 4/10
78/78 [==============================] - 121s 2s/step - loss: 0.3908 - accuracy: 0.8243 - val_loss: 0.6393 - val_accuracy: 0.
Epoch 5/10
78/78 [==============================] - 118s 2s/step - loss: 0.3048 - accuracy: 0.8728 - val_loss: 0.6933 - val_accuracy: 0.
Epoch 6/10
78/78 [==============================] - 119s 2s/step - loss: 0.2332 - accuracy: 0.9011 - val_loss: 0.6327 - val_accuracy: 0.
Epoch 7/10
78/78 [==============================] - 121s 2s/step - loss: 0.1528 - accuracy: 0.9370 - val_loss: 0.8655 - val_accuracy: 0.
Epoch 8/10
78/78 [==============================] - 123s 2s/step - loss: 0.0952 - accuracy: 0.9637 - val_loss: 1.2375 - val_accuracy: 0.
Epoch 9/10
78/78 [==============================] - 120s 2s/step - loss: 0.0658 - accuracy: 0.9737 - val_loss: 1.2063 - val_accuracy: 0.
Epoch 10/10
78/78 [==============================] - 120s 2s/step - loss: 0.0458 - accuracy: 0.9842 - val_loss: 1.4042 - val_accuracy: 0.
22/22 [==============================] - 8s 374ms/step - loss: 1.1097 - accuracy: 0.7645
Test accuracy: 0.7645348906517029
```

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D

# Load the pre-trained MobileNetV2 model without the top (classifier) layers
base_model = MobileNetV2(input_shape=(150, 150, 3), include_top=False, weights='imagenet')

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Create a custom classifier on top of the base model
transfer_model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dense(2, activation='softmax')
])

# Compile the transfer learning model
transfer_model.compile(optimizer=Adam(), loss=SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

# Train the transfer learning model
transfer_model.fit(X_train_normalized, y_train, epochs=10, validation_split=0.1)

# Evaluate the transfer learning model on test data
test_loss, test_accuracy = transfer_model.evaluate(X_test_normalized, y_test)
print(f"Test accuracy: {test_accuracy}")
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_
9406464/9406464 [==============================] - 1s 0us/step
Epoch 1/10
78/78 [==============================] - 57s 688ms/step - loss: 0.1567 - accuracy: 0.9471 - val_loss: 0.1677 - val_accuracy:
Epoch 2/10
78/78 [==============================] - 48s 611ms/step - loss: 0.0709 - accuracy: 0.9721 - val_loss: 0.1724 - val_accuracy:
Epoch 3/10
78/78 [==============================] - 49s 625ms/step - loss: 0.0400 - accuracy: 0.9838 - val_loss: 0.1752 - val_accuracy:
Epoch 4/10
```

```
78/78 [==============================] - 47s 608ms/step - loss: 0.0185 - accuracy: 0.9943 - val_loss: 0.1967 - val_accuracy:
Epoch 5/10
78/78 [==============================] - 45s 584ms/step - loss: 0.0134 - accuracy: 0.9952 - val_loss: 0.2286 - val_accuracy:
Epoch 6/10
78/78 [==============================] - 47s 603ms/step - loss: 0.0079 - accuracy: 0.9984 - val_loss: 0.2152 - val_accuracy:
Epoch 7/10
78/78 [==============================] - 45s 584ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.2236 - val_accuracy:
Epoch 8/10
78/78 [==============================] - 44s 571ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.2430 - val_accuracy:
Epoch 9/10
78/78 [==============================] - 45s 580ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.2338 - val_accuracy:
Epoch 10/10
78/78 [==============================] - 45s 582ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.2486 - val_accuracy:
22/22 [==============================] - 11s 512ms/step - loss: 0.1972 - accuracy: 0.9608
Test accuracy: 0.9607558250427246
```

Simple feedforward neural network:

- Test accuracy: 72.53%

Convolutional neural network (CNN):

- Test accuracy: 75.58%

CNN with more epochs:

- Test accuracy: 76.45%

Transfer learning using MobileNetV2:

- Test accuracy: 96.08%

The best performing model is the transfer learning model using MobileNetV2, which achieved a test accuracy of 96.08%. This model significantly outperforms the other models, demonstrating the benefits of using transfer learning. By leveraging pre-trained weights from a model trained on a large dataset, transfer learning can lead to better feature extraction and classification.