Model_Transfer Learning 01 - No data augmentation - Feature extraction

Name: `Alberto Pingo`
Email: `2202145` @my.ipleiria.pt

Validation dataset: `train5`

Directories

This section sets up the directory paths used for training, validation, and test datasets based on the repository structure.

In [ ]:

```
import os

current_dir = os.getcwd()

# TWO FOLDERS UP
data_dir = os.path.abspath(os.path.join(current_dir, os.pardir, os.pardir
test_dir = os.path.join(data_dir, 'test')
train_dir = os.path.join(data_dir, 'train')

train_dirs = []
for i in range(1, 5):
    train_dirs.append(os.path.join(train_dir, 'train' + str(i)))

validation_dir = os.path.join(data_dir, 'train', 'train5')

print(current_dir)
print(data_dir)
print(test_dir)
print(train_dir)
print(validation_dir)
```

```
/home/pws/code/IA-image-classification/notebooks/models-T
/home/pws/code/IA-image-classification/data
/home/pws/code/IA-image-classification/data/test
/home/pws/code/IA-image-classification/data/train
/home/pws/code/IA-image-classification/data/train/train5
```

Preprocessing

Load the datasets and perform initial preprocessing. Images are resized to 32x32 pixels and batched.

In [ ]:

```
from keras.utils import image_dataset_from_directory
import tensorflow as tf

# Load training datasets from train1 to train4
train_datasets = []
IMG_SIZE = 150
```

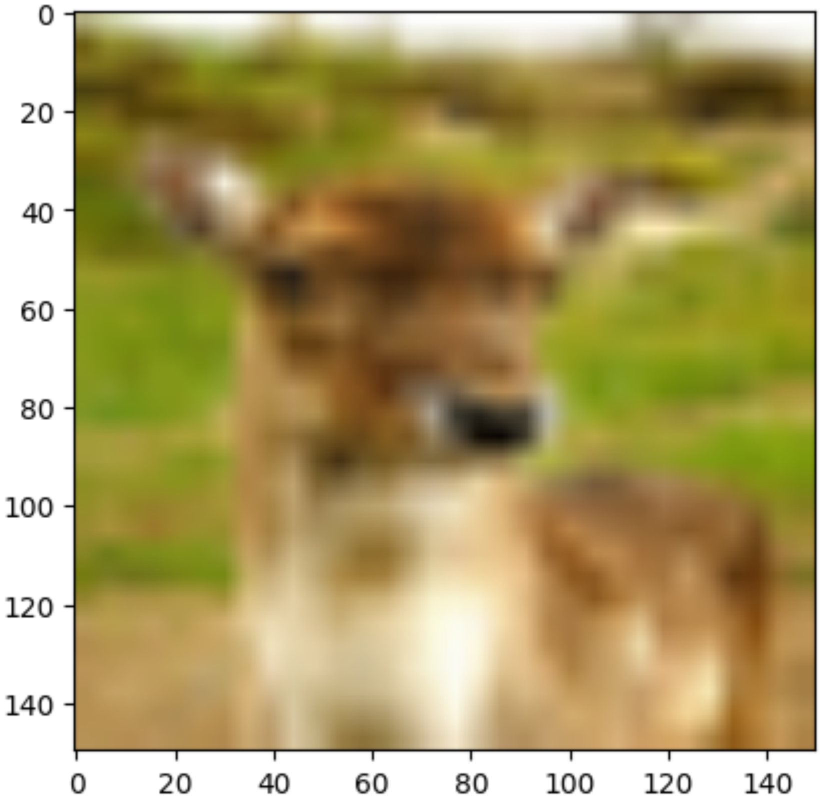
```
BATCH_SIZE = 32

for i in range(1, 5):
    dataset = image_dataset_from_directory(train_dirs[i-1], image_size=(I
    train_datasets.append(dataset)

train_dataset = train_datasets[0]
for dataset in train_datasets[1:]:
    train_dataset = train_dataset.concatenate(dataset)

validation_dataset = image_dataset_from_directory(validation_dir, image_s


test_dataset = image_dataset_from_directory(test_dir, image_size=(IMG_SIZ

class_names = validation_dataset.class_names
class_names = [class_name.split('_')[-1] for class_name in class_names]

print(class_names)
```

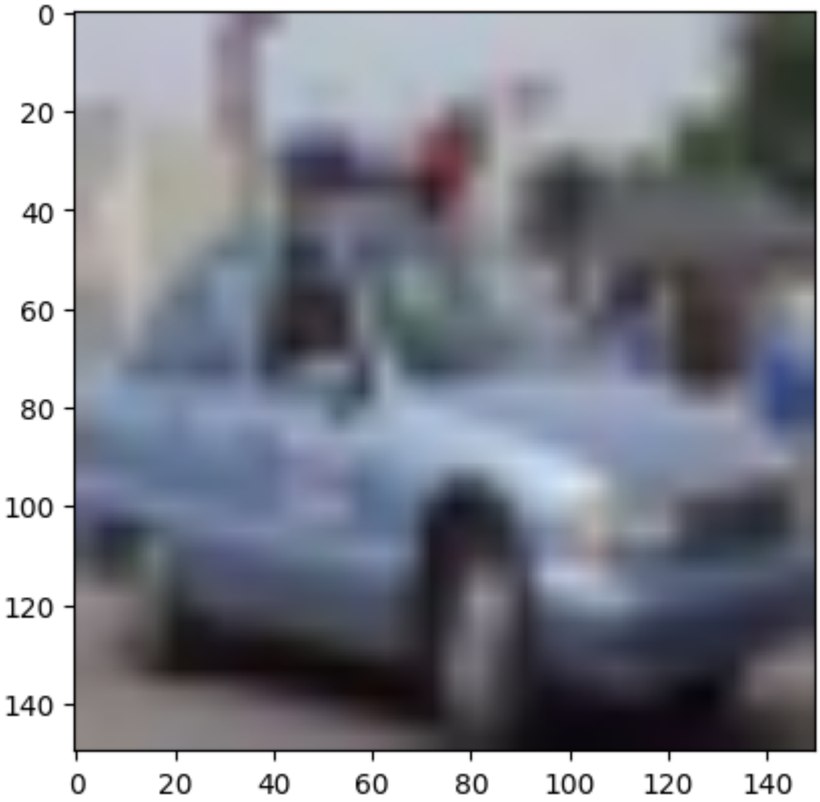
```
2024-06-22 21:29:44.538164: E external/local_xla/xla/stream_executor/cuda/
cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register
factory for plugin cuDNN when one has already been registered
2024-06-22 21:29:44.538220: E external/local_xla/xla/stream_executor/cuda/
cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register
factory for plugin cuFFT when one has already been registered
2024-06-22 21:29:44.632709: E external/local_xla/xla/stream_executor/cuda/
cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to regist
er factory for plugin cuBLAS when one has already been registered
2024-06-22 21:29:44.841271: I tensorflow/core/platform/cpu_feature_guard.c
c:182] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebui
ld TensorFlow with the appropriate compiler flags.
2024-06-22 21:29:46.151776: W tensorflow/compiler/tf2tensorrt/utils/py_uti
ls.cc:38] TF-TRT Warning: Could not find TensorRT
Found 10000 files belonging to 10 classes.
```

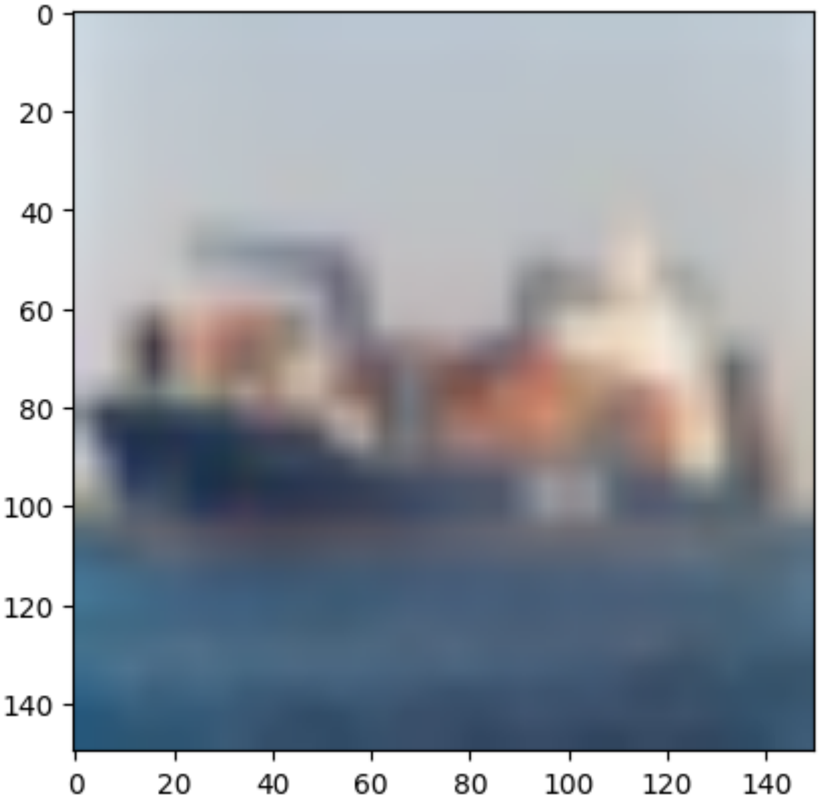
```
2024-06-22 21:29:48.113826: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.487821: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.488018: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.489868: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.490057: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.490209: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.573676: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.573876: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.574048: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-06-22 21:29:48.574145: I tensorflow/core/common_runtime/gpu/gpu_devic
e.cc:1929] Created device /job:localhost/replica:0/task:0/device:GPU:0 wit
h 5070 MB memory:  -> device: 0, name: NVIDIA GeForce GTX 1060 6GB, pci bu
s id: 0000:01:00.0, compute capability: 6.1
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',
'ship', 'truck']
```

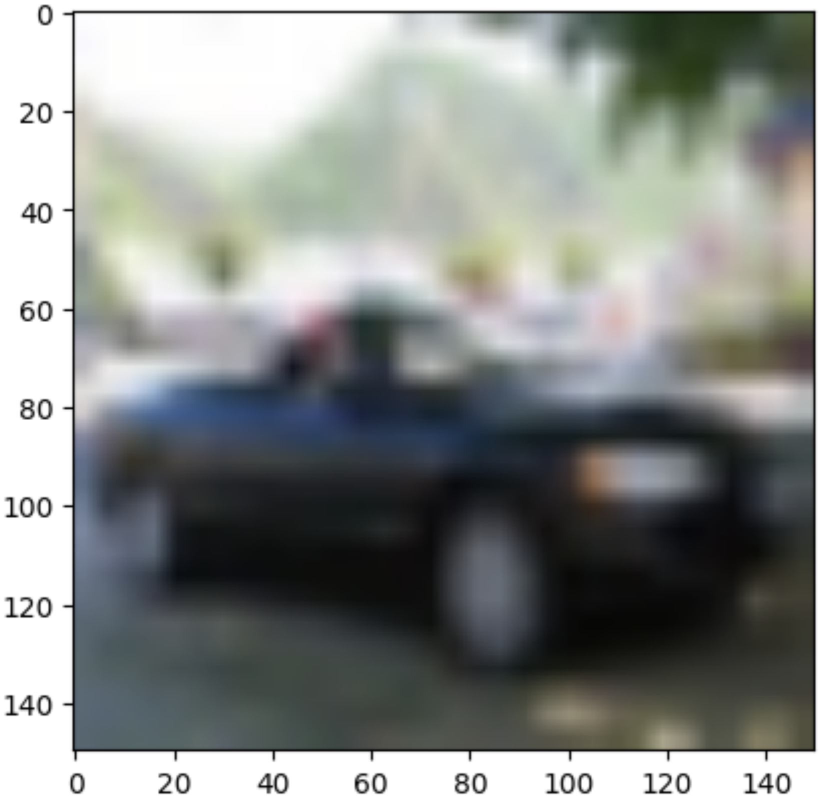
In [ ]:

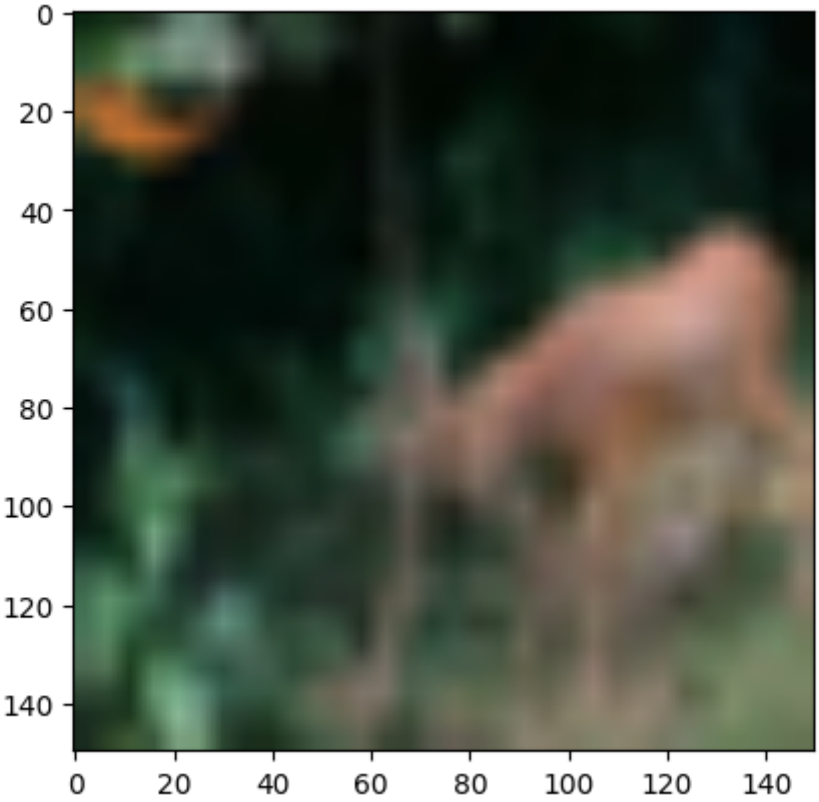
```
import matplotlib.pyplot as plt

for data, _ in train_dataset.take(1):
    for i in range(9):
        plt.imshow(data[i].numpy().astype('uint8'))
        plt.show()
    break
```

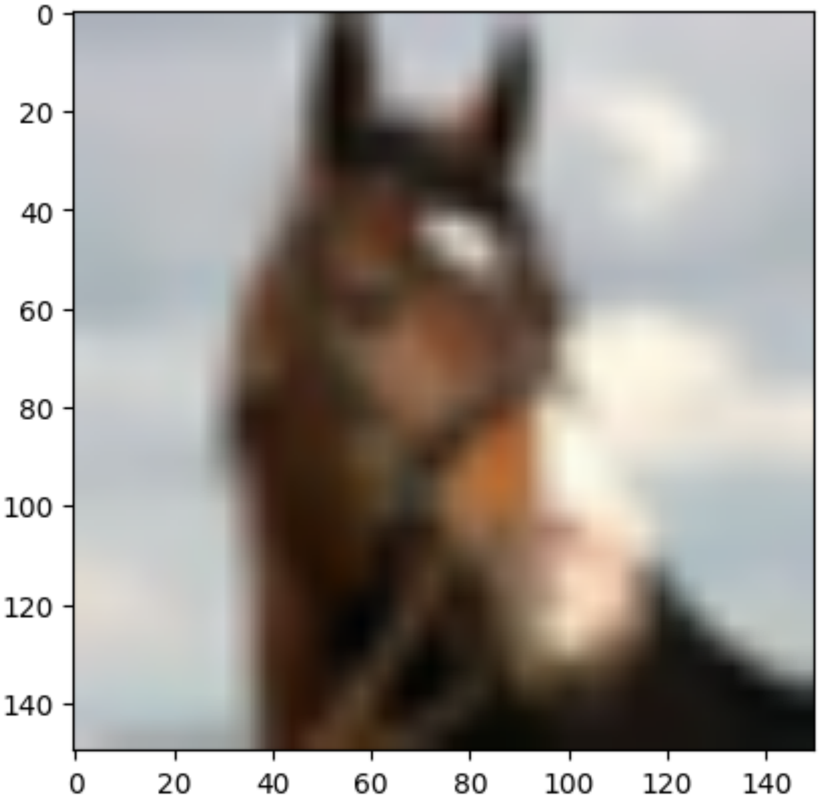


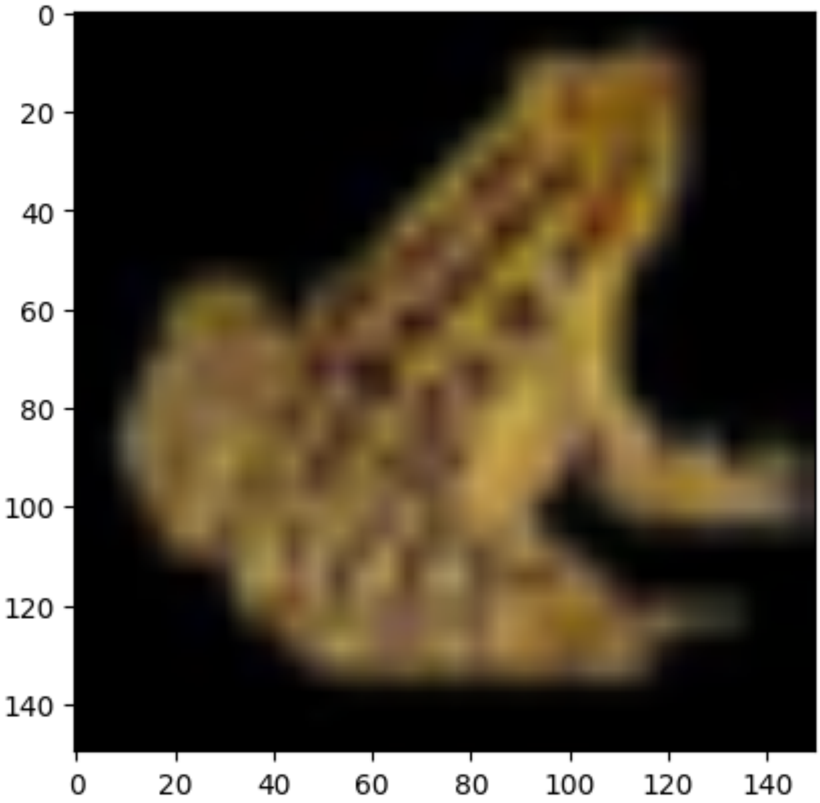


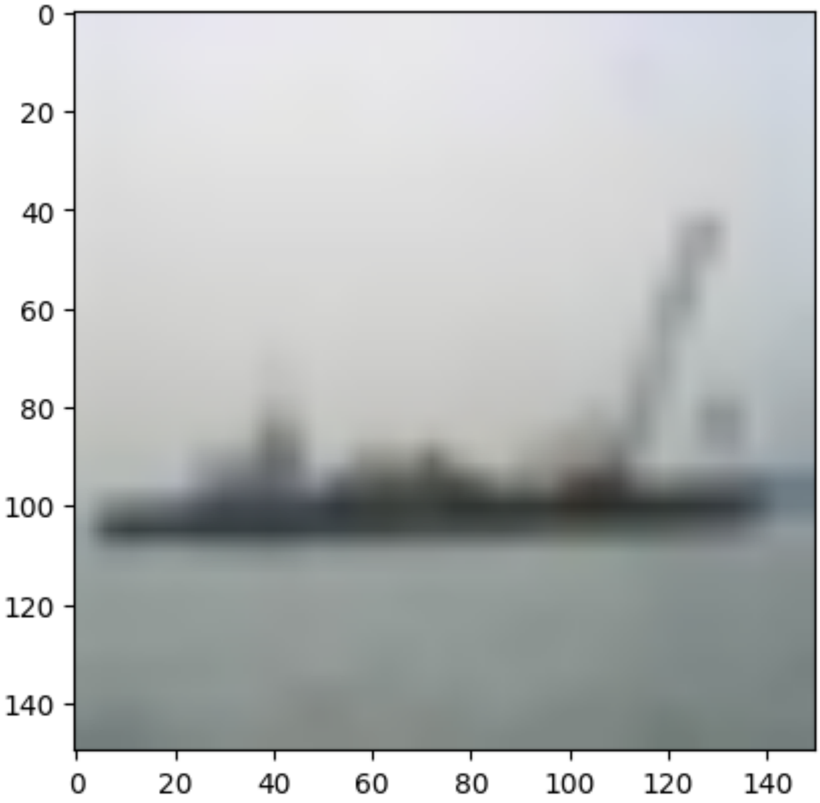

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

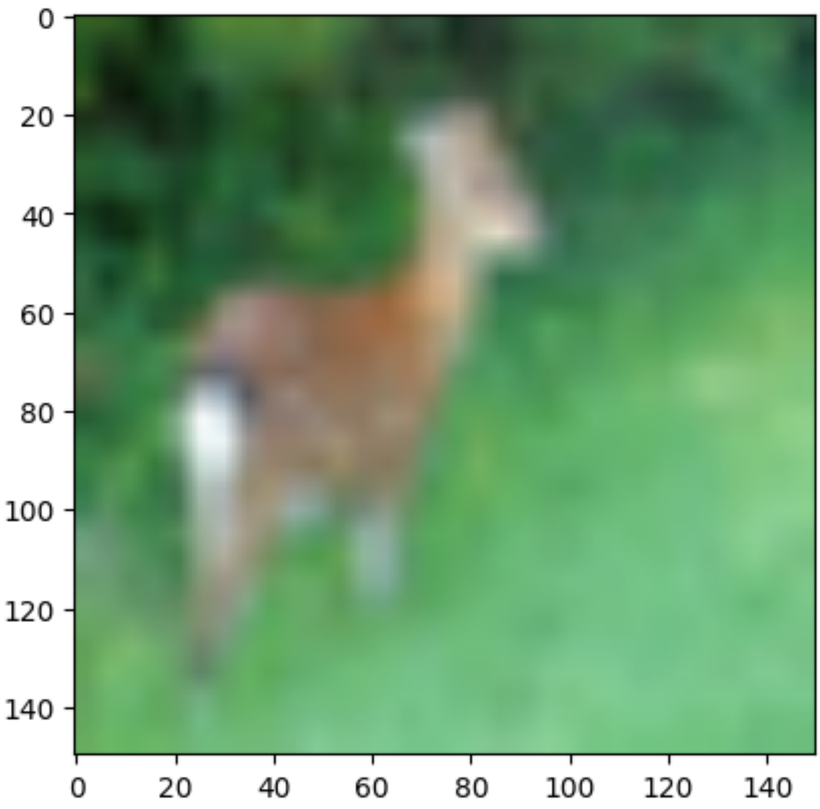

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140



Feature extraction

Use the `VGG16` pre-trained model to extract features from the images.

In [ ]:
```
from tensorflow.keras.applications.vgg16 import VGG16

base_model = VGG16(include_top=False, weights='imagenet', input_shape=(IM
base_model.trainable = False  # Freeze the base model
```

In [ ]:
```
# from tensorflow import keras
# import numpy as np

# def get_features_and_labels(dataset):
#     all_features = []
#     all_labels = []
#     for images, labels in dataset:
#         preprocessed_images = keras.applications.vgg16.preprocess_input
#         features = base_model.predict(preprocessed_images)
#         all_features.append(features)
#         all_labels.append(labels)
#     return np.concatenate(all_features), np.concatenate(all_labels)


# validation_features, validation_labels = get_features_and_labels(valida
# np.save('validation_features.npy', validation_features)
# np.save('validation_labels.npy', validation_labels)
# validation_features = None
# validation_labels = None
# print("Validation features and labels saved")

# test_features, test_labels = get_features_and_labels(test_dataset)
# np.save('test_features.npy', test_features)
# np.save('test_labels.npy', test_labels)
# test_features = None
# test_labels = None
# print("Test features and labels saved")

# train_features, train_labels = get_features_and_labels(train_dataset)
# np.save('train_features.npy', train_features)
# np.save('train_labels.npy', train_labels)
# train_features = None
# train_labels = None
# print("Train features and labels saved")
```

Load the extracted features

In [ ]:

```
from numpy import load

train_features = load('features_T/train_features.npy')
train_labels = load('features_T/train_labels.npy')

validation_features = load('features_T/validation_features.npy')
validation_labels = load('features_T/validation_labels.npy')

test_features = load('features_T/test_features.npy')
test_labels = load('features_T/test_labels.npy')
```

MODEL ARCHITECTURE

Transfer Learning Model

Use the extracted features to train a model for the classification task.

Use two dense layers with 256 neurons and a in-between dropout layer with a rate of 0.5.

In [ ]:

```
from tensorflow import keras
from keras.applications import VGG16
from keras import layers

inputs = keras.Input(shape=(4, 4, 512))

x = layers.Flatten()(inputs)
x = layers.Dense(256, activation="relu")(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(256, activation="relu")(x)
x = layers.Dropout(0.5)(x)

outputs = layers.Dense(10, activation="softmax")(x)  # Softmax for multi-

model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 4, 4, 512)]       0         
                                                                 
 flatten (Flatten)           (None, 8192)              0         
                                                                 
 dense (Dense)               (None, 256)               2097408   
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 dense_1 (Dense)             (None, 256)               65792     
                                                                 
 dropout_1 (Dropout)         (None, 256)               0         
                                                                 
 dense_2 (Dense)             (None, 10)                2570      
                                                                 
=================================================================
Total params: 2165770 (8.26 MB)
Trainable params: 2165770 (8.26 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Compile Model

Loss function:
We use the *Categorical Crossentropy* loss function because it is a `multi-class classification` problem.

Optimizer: Adam
We use the *Adam* optimizer because it is one of the best and most popular optimizers.

In [ ]:

```
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['acc'])
```

Train Model

Train the model with Early stopping, Model checkpoint, and Learning rate reduction callbacks.

In [ ]:

```
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPla

learning_rate_reduction = ReduceLROnPlateau(
    monitor='val_acc',
    patience=3,
    verbose=1,
    factor=0.5,
    min_lr=1e-6)

early_stop = EarlyStopping(monitor='val_acc',
                           patience=5,
                           restore_best_weights=True)
```

```python
model_checkpoint = ModelCheckpoint('models/T01/checkpoints/T01-cp.h5', sa

history = model.fit(
    train_features, train_labels,
    epochs=50,
    validation_data=(validation_features, validation_labels),
    callbacks=[early_stop, model_checkpoint, learning_rate_reduction])
```

```
2024-06-22 21:29:59.800833: W external/local_tsl/tsl/framework/cpu_allocat
or_impl.cc:83] Allocation of 1310720000 exceeds 10% of free system memory.
2024-06-22 21:30:00.629393: W external/local_tsl/tsl/framework/cpu_allocat
or_impl.cc:83] Allocation of 1310720000 exceeds 10% of free system memory.
Epoch 1/50
```

```
2024-06-22 21:30:02.762521: I external/local_xla/xla/service/service.cc:16
8] XLA service 0x7db791188ae0 initialized for platform CUDA (this does not
guarantee that XLA will be used). Devices:
2024-06-22 21:30:02.762543: I external/local_xla/xla/service/service.cc:17
6]   StreamExecutor device (0): NVIDIA GeForce GTX 1060 6GB, Compute Capab
ility 6.1
2024-06-22 21:30:02.793368: I tensorflow/compiler/mlir/tensorflow/utils/du
mp_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CR
ASH_REPRODUCER_DIRECTORY` to enable.
2024-06-22 21:30:02.858548: I external/local_xla/xla/stream_executor/cuda/
cuda_dnn.cc:454] Loaded cuDNN version 8904
WARNING: All log messages before absl::InitializeLog() is called are writt
en to STDERR
I0000 00:00:1719088202.944203    4013 device_compiler.h:186] Compiled clus
ter using XLA!  This line is logged at most once for the lifetime of the p
rocess.
1250/1250 [==============================] - 7s 4ms/step - loss: 1.6091 -
acc: 0.5875 - val_loss: 0.8074 - val_acc: 0.7531 - lr: 0.0010
Epoch 2/50
  20/1250 [..............................] - ETA: 3s - loss: 0.9607 - acc:
0.6906
```

```
/home/pws/miniconda3/envs/tensorflow/lib/python3.11/site-packages/keras/sr
c/engine/training.py:3103: UserWarning: You are saving your model as an HD
F5 file via `model.save()`. This file format is considered legacy. We reco
mmend using instead the native Keras format, e.g. `model.save('my_model.ke
ras')`.
  saving_api.save_model(
1250/1250 [==============================] - 4s 4ms/step - loss: 0.9766 -
acc: 0.7104 - val_loss: 0.5722 - val_acc: 0.8305 - lr: 0.0010
Epoch 3/50
1250/1250 [==============================] - 5s 4ms/step - loss: 0.7366 -
acc: 0.7757 - val_loss: 0.5336 - val_acc: 0.8462 - lr: 0.0010
Epoch 4/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.6321 -
acc: 0.8100 - val_loss: 0.4832 - val_acc: 0.8607 - lr: 0.0010
Epoch 5/50
1250/1250 [==============================] - 5s 4ms/step - loss: 0.5985 -
acc: 0.8208 - val_loss: 0.4522 - val_acc: 0.8618 - lr: 0.0010
Epoch 6/50
1250/1250 [==============================] - 5s 4ms/step - loss: 0.5567 -
acc: 0.8350 - val_loss: 0.4580 - val_acc: 0.8680 - lr: 0.0010
Epoch 7/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.5391 -
acc: 0.8418 - val_loss: 0.4585 - val_acc: 0.8613 - lr: 0.0010
Epoch 8/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.4967 -
acc: 0.8530 - val_loss: 0.4723 - val_acc: 0.8649 - lr: 0.0010
Epoch 9/50
1240/1250 [============================>.] - ETA: 0s - loss: 0.5026 - acc:
0.8539
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.000500000023748725
7.
1250/1250 [==============================] - 4s 3ms/step - loss: 0.5028 -
acc: 0.8538 - val_loss: 0.4795 - val_acc: 0.8632 - lr: 0.0010
Epoch 10/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.4069 -
acc: 0.8728 - val_loss: 0.4460 - val_acc: 0.8729 - lr: 5.0000e-04
Epoch 11/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.3741 -
acc: 0.8830 - val_loss: 0.4377 - val_acc: 0.8785 - lr: 5.0000e-04
Epoch 12/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.3625 -
acc: 0.8867 - val_loss: 0.4363 - val_acc: 0.8715 - lr: 5.0000e-04
Epoch 13/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.3295 -
acc: 0.8942 - val_loss: 0.4495 - val_acc: 0.8750 - lr: 5.0000e-04
Epoch 14/50
1237/1250 [============================>.] - ETA: 0s - loss: 0.3203 - acc:
0.8985
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.000250000011874362
8.
1250/1250 [==============================] - 4s 3ms/step - loss: 0.3201 -
acc: 0.8985 - val_loss: 0.4594 - val_acc: 0.8736 - lr: 5.0000e-04
Epoch 15/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.2963 -
acc: 0.9048 - val_loss: 0.4596 - val_acc: 0.8749 - lr: 2.5000e-04
Epoch 16/50
1250/1250 [==============================] - 4s 3ms/step - loss: 0.2694 -
acc: 0.9103 - val_loss: 0.4673 - val_acc: 0.8781 - lr: 2.5000e-04
```

Save Model

In [ ]:

```
keras.models.save_model(model, 'models/T01/T01-model.h5')
```

```
/tmp/ipykernel_3734/1570624933.py:1: UserWarning: You are saving your mode
l as an HDF5 file via `model.save()`. This file format is considered legac
y. We recommend using instead the native Keras format, e.g. `model.save('m
y_model.keras')`.
  keras.models.save_model(model, 'models/T01/T01-model.h5')
```

Load Model

In [ ]:

```python
keras.models.load_model('models/T01/T01-model.h5')
```

Out[ ]:

```
<keras.src.engine.functional.Functional at 0x7db834684110>
```

EVALUATION

Evaluate the model on the validation dataset.

In [ ]:

```python
val_loss, val_acc = model.evaluate(validation_features, validation_labels
print('val_acc:', val_acc)
```

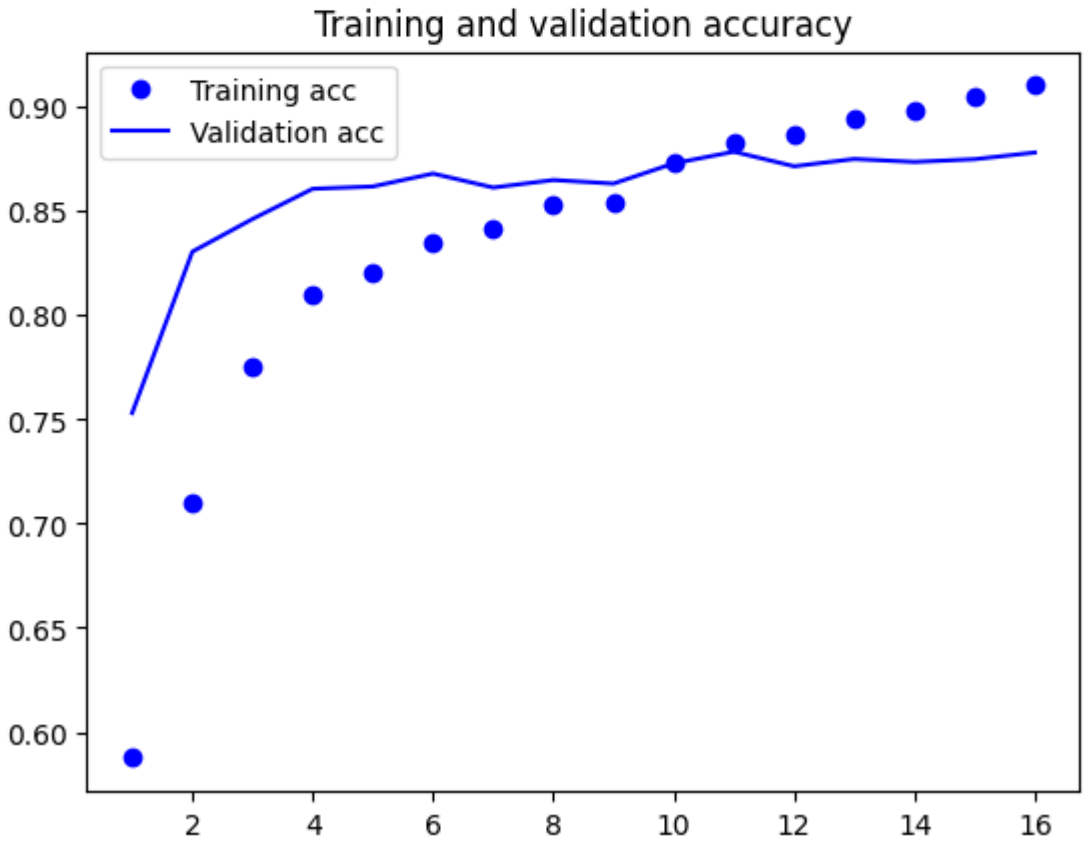
```
313/313 [==============================] - 1s 2ms/step - loss: 0.4377 - ac
c: 0.8785
val_acc: 0.8784999847412109
```

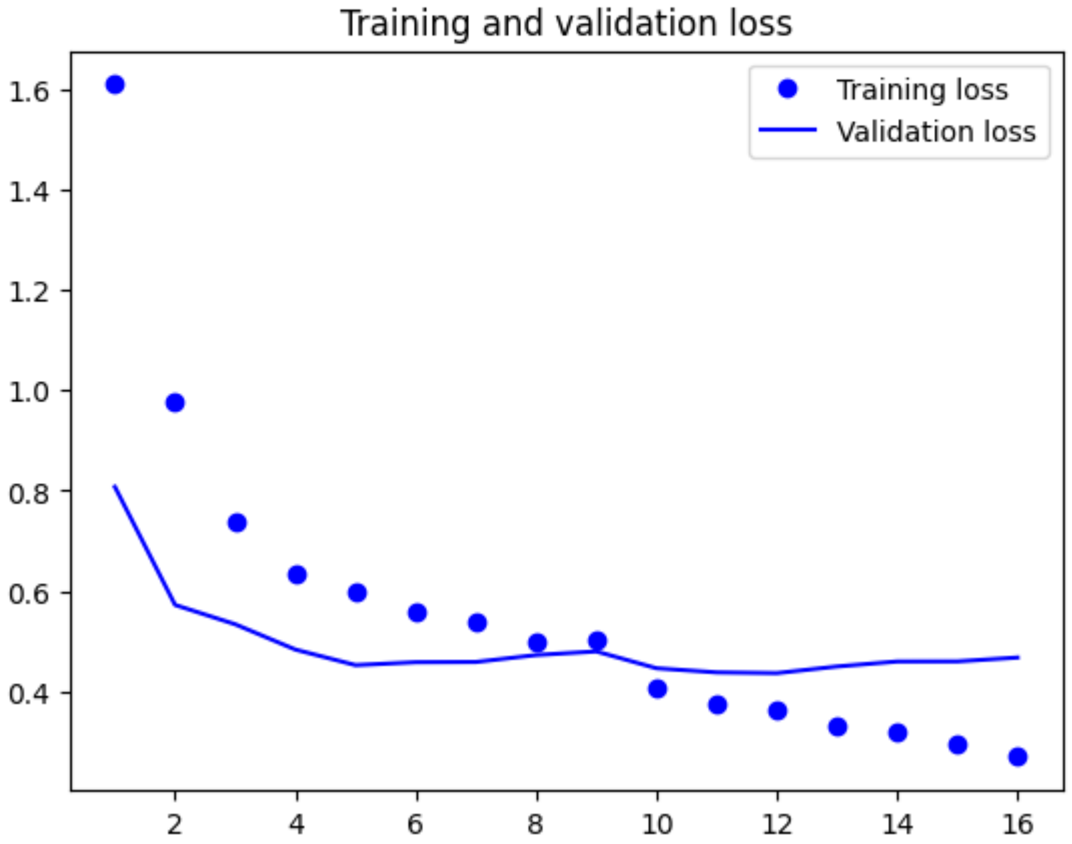
Training and Validation Curves

Plot the training and validation accuracy and loss curves.

In [ ]:

```python
import matplotlib.pyplot as plt

# Extract the history from the training process
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

# Plot the training and validation accuracy
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

# Plot the training and validation loss
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





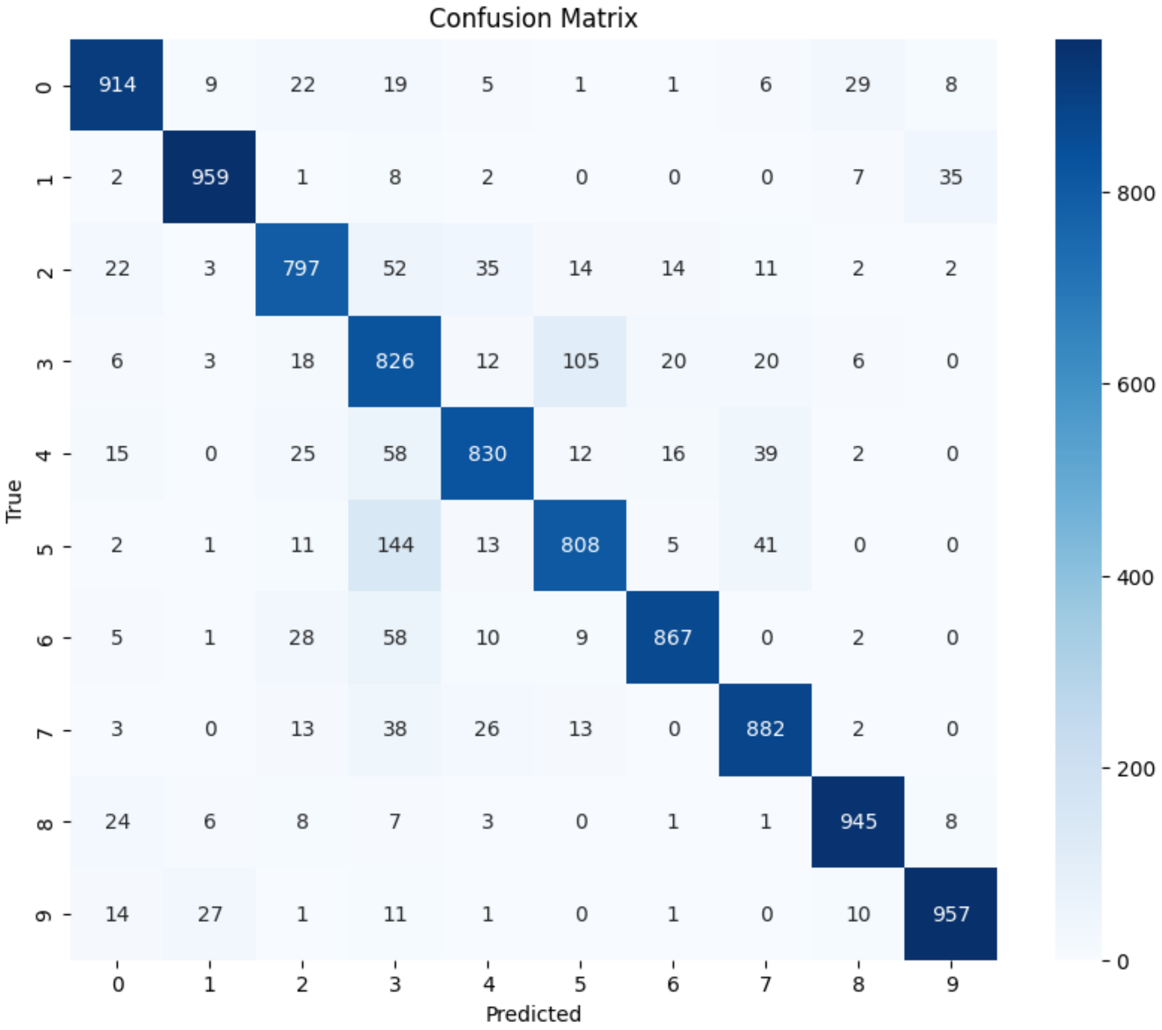
Confusion Matrix

In [ ]:

```
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
y_pred = np.argmax(model.predict(validation_features), axis=1)
y_true = np.argmax(validation_labels, axis=1)

cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
313/313 [==============================] - 0s 1ms/step
```



	0	1	2	3	4	5	6	7	8	9
0	914	9	22	19	5	1	1	6	29	8
1	2	959	1	8	2	0	0	0	7	35
2	22	3	797	52	35	14	14	11	2	2
3	6	3	18	826	12	105	20	20	6	0
4	15	0	25	58	830	12	16	39	2	0
5	2	1	11	144	13	808	5	41	0	0
6	5	1	28	58	10	9	867	0	2	0
7	3	0	13	38	26	13	0	882	2	0
8	24	6	8	7	3	0	1	1	945	8
9	14	27	1	11	1	0	1	0	10	957

In [ ]:

```
from sklearn.metrics import classification_report

report = classification_report(y_true, y_pred, target_names=class_names)
print(report)
```

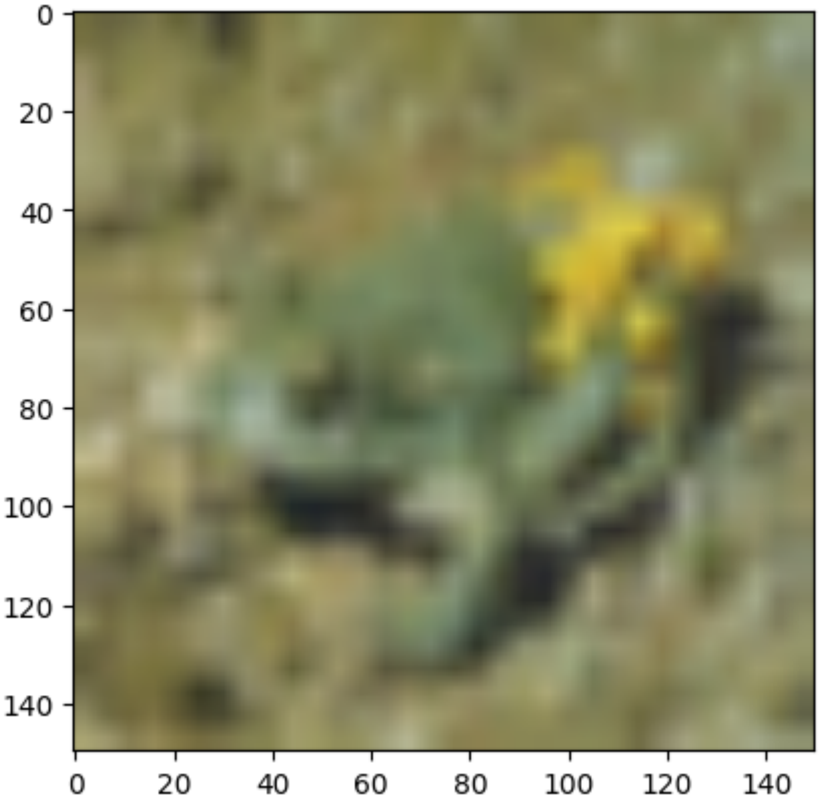
```
              precision    recall  f1-score   support

    airplane       0.91      0.90      0.90      1014
  automobile       0.95      0.95      0.95      1014
        bird       0.86      0.84      0.85       952
         cat       0.68      0.81      0.74      1016
        deer       0.89      0.83      0.86       997
         dog       0.84      0.79      0.81      1025
        frog       0.94      0.88      0.91       980
       horse       0.88      0.90      0.89       977
        ship       0.94      0.94      0.94      1003
       truck       0.95      0.94      0.94      1022

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```

Predictions

Predict and visualize the results for a sample image.

In [ ]:

```python
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.preprocessing import image
from keras.applications.vgg16 import VGG16, preprocess_input
import numpy as np

# Load an image
img_path = train_dirs[0] + '/006_frog/alytes_obstetricans_s_000179.png'
img = tf.keras.preprocessing.image.load_img(img_path, target_size=(150, 1

# Preprocess the image for VGG16
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

plt.imshow(img)
plt.show()

print(img_array.shape)

base_model = VGG16(weights='imagenet', include_top=False, input_shape=(15

# Extract features using VGG16
features = base_model.predict(img_array)

flattened_features = features.reshape((features.shape[0], -1))

# Predict using your custom model
result = model.predict(flattened_features)

print("Result: ", result.round())
print("Predicted class: ", class_names[np.argmax(result)])
print("True class: ", img_path.split('/')[-2].split('_')[-1])
```



```
(1, 150, 150, 3)
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 67ms/step
Result:  [[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]]
Predicted class:  frog
True class:  frog
```