In [1]:
```
#@title Kaggle data import

#from google.colab import files
#files.upload()
#!mkdir ~/.kaggle
#!cp kaggle.json ~/.kaggle/
#!chmod 600 ~/.kaggle/kaggle.json

#import kagglehub

#dataset for a convolutional Neural Network
CNN_path = 'animal_data'

#dataset for classification model
#Class_path = kagglehub.dataset_download("ehababoelnaga/multiple-disease-prediction")
```

In [2]:
```
CNN_path
```

Out[2]:
```
'animal_data'
```

In [3]:
```
!pip install keras
!pip install tensorflow
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: keras in c:\users\lcleymaet\appdata\roaming\python\python312\site-
packages (3.9.2)
Requirement already satisfied: absl-py in c:\users\lcleymaet\appdata\roaming\python\python312\sit
e-packages (from keras) (2.2.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from keras)
(1.26.4)
Requirement already satisfied: rich in c:\programdata\anaconda3\lib\site-packages (from keras) (1
3.9.4)
Requirement already satisfied: namex in c:\users\lcleymaet\appdata\roaming\python\python312\site-
packages (from keras) (0.0.9)
Requirement already satisfied: h5py in c:\programdata\anaconda3\lib\site-packages (from keras)
(3.12.1)
Requirement already satisfied: optree in c:\users\lcleymaet\appdata\roaming\python\python312\site
-packages (from keras) (0.15.0)
Requirement already satisfied: ml-dtypes in c:\users\lcleymaet\appdata\roaming\python\python312\s
ite-packages (from keras) (0.5.1)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from kera
s) (24.2)
Requirement already satisfied: typing-extensions>=4.5.0 in c:\programdata\anaconda3\lib\site-pack
ages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\programdata\anaconda3\lib\site-package
s (from rich->keras) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\programdata\anaconda3\lib\site-packa
ges (from rich->keras) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\programdata\anaconda3\lib\site-packages (from mar
kdown-it-py>=2.2.0->rich->keras) (0.1.0)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow in c:\users\lcleymaet\appdata\roaming\python\python312
\site-packages (2.19.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\lcleymaet\appdata\roaming\python\python
312\site-packages (from tensorflow) (2.2.2)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\lcleymaet\appdata\roaming\python\pyt
hon312\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\lcleymaet\appdata\roaming\python
\python312\site-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\lcleymaet\appdata
\roaming\python\python312\site-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\lcleymaet\appdata\roaming\python\p
ython312\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\lcleymaet\appdata\roaming\python\pyth
on312\site-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\lcleymaet\appdata\roaming\python\pyt
hon312\site-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from tens
orflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.
0.0dev,>=3.20.3 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda3\lib\site-packages
(from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from ten
sorflow) (75.8.0)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\site-packages (from te
nsorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\lcleymaet\appdata\roaming\python\pyth
on312\site-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\anaconda3\lib\site-pack
ages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\lib\site-packages (from
```

```
tensorflow) (1.17.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\lcleymaet\appdata\roaming\python\p
ython312\site-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard~=2.19.0 in c:\users\lcleymaet\appdata\roaming\python\p
ython312\site-packages (from tensorflow) (2.19.0)
Requirement already satisfied: keras>=3.5.0 in c:\users\lcleymaet\appdata\roaming\python\python31
2\site-packages (from tensorflow) (3.9.2)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in c:\programdata\anaconda3\lib\site-packages
(from tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in c:\programdata\anaconda3\lib\site-packages (from t
ensorflow) (3.12.1)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in c:\users\lcleymaet\appdata\roaming\pyth
on\python312\site-packages (from tensorflow) (0.5.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\programdata\anaconda3\lib\site-packages
(from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: rich in c:\programdata\anaconda3\lib\site-packages (from keras>=3.
5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in c:\users\lcleymaet\appdata\roaming\python\python312\site-
packages (from keras>=3.5.0->tensorflow) (0.0.9)
Requirement already satisfied: optree in c:\users\lcleymaet\appdata\roaming\python\python312\site
-packages (from keras>=3.5.0->tensorflow) (0.15.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-pack
ages (from requests<3,>=2.21.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from r
equests<3,>=2.21.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages
(from requests<3,>=2.21.0->tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages
(from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\site-packages (fro
m tensorboard~=2.19.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\lcleymaet\appdat
a\roaming\python\python312\site-packages (from tensorboard~=2.19.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (fro
m tensorboard~=2.19.0->tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\programdata\anaconda3\lib\site-packages (f
rom werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\programdata\anaconda3\lib\site-package
s (from rich->keras>=3.5.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\programdata\anaconda3\lib\site-packa
ges (from rich->keras>=3.5.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\programdata\anaconda3\lib\site-packages (from mar
kdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.0)
```

In [28]:
```
import sys
print(sys.executable)
```

```
C:\ProgramData\anaconda3\python.exe
```

In [5]:
```
!{sys.executable} -m pip install tensorflow keras
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow in c:\users\lcleymaet\appdata\roaming\python\python312
\site-packages (2.19.0)
Requirement already satisfied: keras in c:\users\lcleymaet\appdata\roaming\python\python312\site-
packages (3.9.2)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\lcleymaet\appdata\roaming\python\python
312\site-packages (from tensorflow) (2.2.2)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\lcleymaet\appdata\roaming\python\pyt
hon312\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\lcleymaet\appdata\roaming\python
\python312\site-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\lcleymaet\appdata
\roaming\python\python312\site-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\lcleymaet\appdata\roaming\python\p
ython312\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\lcleymaet\appdata\roaming\python\pyth
on312\site-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\lcleymaet\appdata\roaming\python\pyt
hon312\site-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from tens
orflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.
0.0dev,>=3.20.3 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda3\lib\site-packages
(from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from ten
sorflow) (75.8.0)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\site-packages (from te
nsorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\lcleymaet\appdata\roaming\python\pyth
on312\site-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\anaconda3\lib\site-pack
ages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\lib\site-packages (from
tensorflow) (1.17.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\lcleymaet\appdata\roaming\python\p
ython312\site-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard~=2.19.0 in c:\users\lcleymaet\appdata\roaming\python\p
ython312\site-packages (from tensorflow) (2.19.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in c:\programdata\anaconda3\lib\site-packages
(from tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in c:\programdata\anaconda3\lib\site-packages (from t
ensorflow) (3.12.1)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in c:\users\lcleymaet\appdata\roaming\pyth
on\python312\site-packages (from tensorflow) (0.5.1)
Requirement already satisfied: rich in c:\programdata\anaconda3\lib\site-packages (from keras) (1
3.9.4)
Requirement already satisfied: namex in c:\users\lcleymaet\appdata\roaming\python\python312\site-
packages (from keras) (0.0.9)
Requirement already satisfied: optree in c:\users\lcleymaet\appdata\roaming\python\python312\site
-packages (from keras) (0.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\programdata\anaconda3\lib\site-packages
(from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-pack
ages (from requests<3,>=2.21.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from r
equests<3,>=2.21.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages
(from requests<3,>=2.21.0->tensorflow) (2.3.0)
```

```
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages
(from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\site-packages (fro
m tensorboard~=2.19.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\lcleymaet\appdat
a\roaming\python\python312\site-packages (from tensorboard~=2.19.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (fro
m tensorboard~=2.19.0->tensorflow) (3.1.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\programdata\anaconda3\lib\site-package
s (from rich->keras) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\programdata\anaconda3\lib\site-packa
ges (from rich->keras) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\programdata\anaconda3\lib\site-packages (from mar
kdown-it-py>=2.2.0->rich->keras) (0.1.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\programdata\anaconda3\lib\site-packages (f
rom werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (3.0.2)
```

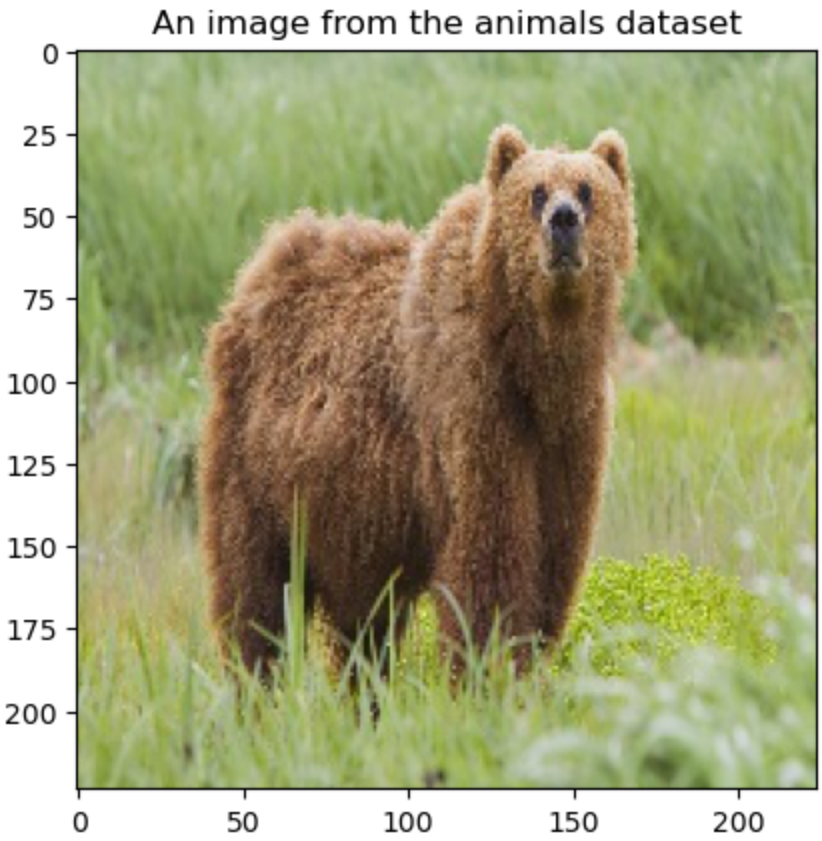
In [6]:
```
#@title CNN data preparation

import os
import numpy as np
import pandas as pd
import tensorflow as ts
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
import pprint

from keras.preprocessing import image_dataset_from_directory
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, InputLayer, AveragePooli
from keras import regularizers
from sklearn.metrics import confusion_matrix

from keras.utils import to_categorical
from sklearn.utils import shuffle

#build dictionary for animal types
animals = {0:'Bear',1:'Bird',2:'Cat',3:'Cow',4:'Deer',5:'Dog',6:'Dolphin',7:'Elephant',8:'Giraff
subdirs = list(animals.values())
```

In [7]:
```
#Check image sizes
from PIL import Image

shapes = []

for subdir in os.listdir(CNN_path):
    temp_path = os.path.join(CNN_path, subdir)
    if not os.path.isdir(temp_path):
        continue
    for fn in os.listdir(temp_path):
        im_path = os.path.join(temp_path, fn)
        img = Image.open(im_path)
        if img.size not in shapes:
            shapes.append(img.size)

print(f"The images have shapes {shapes}\n")
```

```
The images have shapes [(224, 224)]
```

In [8]:
```
#@title checking if the images are color or just rgb grayscale

path = CNN_path+'/'+subdirs[0]+'/'
file_path = path+os.listdir(path)[0]
img = mpimg.imread(file_path)
plt.title('An image from the animals dataset')
plt.imshow(img)
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x1b6a5524da0>
```



All of the images in this dataset are of the same shape so no need to resize. It looks like color will matter so we will not flatten it to grayscale, at least to start. If resizing was necessary, I would implement the following code:

```
from PIL import Image

#create new directory to store resized images to preserve original dataset
for subdir in subdirs:
  path = CNN_path + '/' + subdir + '/'
  newpath = CNN_path + '/' + subdir + '_new/'

  os.makedirs(path_new, exist_ok = True)

  for fn in os.listdir(path):
    img = Image.open(path + fn)
    img_new = img.resize((224, 224))
    img_new.save(newpath + fn)
```

Since we don't need that, we will proceed with importing the dataset.

In [9]:
```
img_size = (224,224)

data = image_dataset_from_directory(
    CNN_path,
    labels = 'inferred',
    label_mode = 'categorical',
    image_size=img_size,
    batch_size=32,
    shuffle = True
)
```

```
Found 1944 files belonging to 15 classes.
```

In [10]:
```
#prepare dataset for model

X = []
Y = []

for batch in data:
    im, label = batch
    X.append(im)
    Y.append(label)

X = np.concatenate(X)
Y = np.concatenate(Y)

print(f"X shape is {X.shape}")
print(f"Y shape is {Y.shape}")

X_shape = X.shape
Y_shape = Y.shape
```

```
X shape is (1944, 224, 224, 3)
Y shape is (1944, 15)
```

In [11]:
```
#normalize X
X = X / 255

#split dataset
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3, random_state = 2525)
```

In [21]:
```
#start with a simple model

animal_CNN_1 = Sequential()
animal_CNN_1.add(InputLayer(input_shape = X_shape[1:]))
animal_CNN_1.add(Conv2D(filters = 32,
                        kernel_size = (3, 3),
                        activation = 'relu',
                        padding = 'same'))
animal_CNN_1.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_1.add(Dropout(0.5))

animal_CNN_1.add(Conv2D(filters = 64,
                        kernel_size = (3, 3),
                        padding = 'same',
                        activation = 'relu'))
animal_CNN_1.add(MaxPooling2D(pool_size = (2,2)))
```

```
animal_CNN_1.add(Dropout(0.5))

animal_CNN_1.add(Flatten())
animal_CNN_1.add(Dense(128, activation = 'relu'))
animal_CNN_1.add(Dropout(0.25))

animal_CNN_1.add(Dense(Y_shape[1], activation = 'softmax'))

animal_CNN_1.compile(loss = 'categorical_crossentropy',
                     optimizer = 'adam',
                     metrics = ['accuracy'])

animal_CNN_1.summary()
```

```
C:\Users\lcleymaet\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\core\input_layer.py:27: UserWarning: Argument `input_shape` is deprecated. Use `shape` instead.
  warnings.warn(
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
dropout (Dropout)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_1 (Dropout)	(None, 56, 56, 64)	0
flatten (Flatten)	(None, 200704)	0
dense (Dense)	(None, 128)	25,690,240
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 15)	1,935

Total params: 25,711,567 (98.08 MB)

Trainable params: 25,711,567 (98.08 MB)

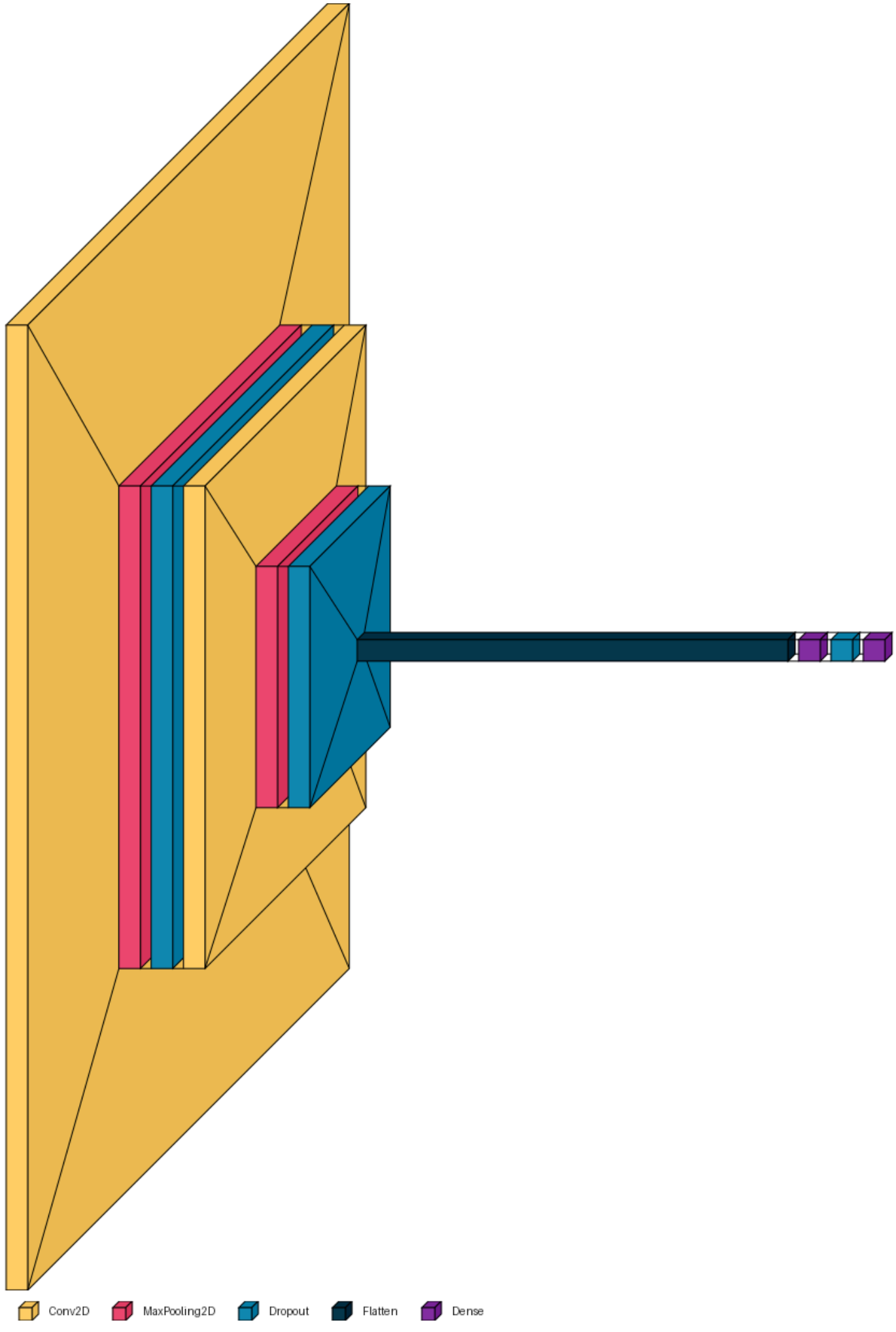
Non-trainable params: 0 (0.00 B)

Let's visualize this model for fun.

In [23]:
```
!pip install visualkeras
import visualkeras

visualkeras.layered_view(animal_CNN_1, legend = True)
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting visualkeras
  Downloading visualkeras-0.1.4-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from
visualkeras) (11.1.0)
Requirement already satisfied: numpy>=1.18.1 in c:\programdata\anaconda3\lib\site-packages (from
visualkeras) (1.26.4)
Collecting aggdraw>=1.3.11 (from visualkeras)
  Downloading aggdraw-1.3.19-cp312-cp312-win_amd64.whl.metadata (673 bytes)
Downloading visualkeras-0.1.4-py3-none-any.whl (17 kB)
Downloading aggdraw-1.3.19-cp312-cp312-win_amd64.whl (45 kB)
Installing collected packages: aggdraw, visualkeras
Successfully installed aggdraw-1.3.19 visualkeras-0.1.4
C:\Users\lcleymaet\AppData\Roaming\Python\Python312\site-packages\visualkeras\layered.py:86: User
Warning: The legend_text_spacing_offset parameter is deprecated and will be removed in a future r
elease.
  warnings.warn("The legend_text_spacing_offset parameter is deprecated and will be removed in a
future release.")
```

Out[23]:



In [24]:

```
#@title Training the model

#set hyperparams
epochs = 50
batch_size = 112

#fit model, including validation monitoring to help prevent overfitting
history_CNN_1 = animal_CNN_1.fit(X_train,
                                 Y_train,
                                 epochs = epochs,
                                 batch_size = batch_size,
                                validation_split = 0.2)
```

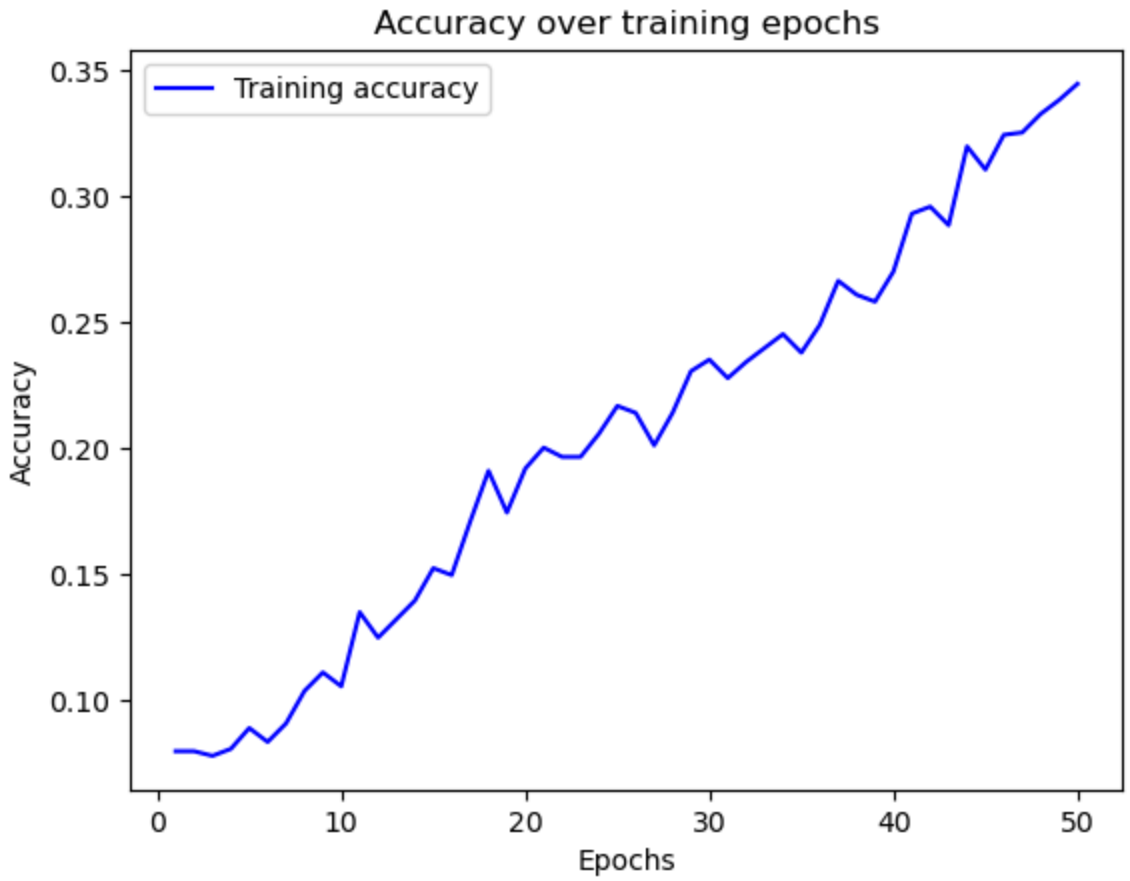
```
Epoch 1/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 12s 601ms/step - accuracy: 0.0864 - loss: 14.0189 - val_accuracy: 0.08
46 - val_loss: 2.7054
Epoch 2/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 500ms/step - accuracy: 0.0697 - loss: 2.7192 - val_accuracy: 0.0772
- val_loss: 2.7079
Epoch 3/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 490ms/step - accuracy: 0.0866 - loss: 2.7009 - val_accuracy: 0.0809
- val_loss: 2.7078
Epoch 4/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 488ms/step - accuracy: 0.0819 - loss: 2.7010 - val_accuracy: 0.0735
- val_loss: 2.7077
Epoch 5/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 496ms/step - accuracy: 0.0773 - loss: 2.7002 - val_accuracy: 0.0478
- val_loss: 2.7078
Epoch 6/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 485ms/step - accuracy: 0.0839 - loss: 2.6923 - val_accuracy: 0.1029
- val_loss: 2.7063
Epoch 7/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 486ms/step - accuracy: 0.0905 - loss: 2.6637 - val_accuracy: 0.0735
- val_loss: 2.7064
Epoch 8/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 487ms/step - accuracy: 0.1006 - loss: 2.6829 - val_accuracy: 0.0735
- val_loss: 2.6922
Epoch 9/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 490ms/step - accuracy: 0.0987 - loss: 2.6481 - val_accuracy: 0.0809
- val_loss: 2.6872
Epoch 10/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 494ms/step - accuracy: 0.1098 - loss: 2.6362 - val_accuracy: 0.0993
- val_loss: 2.6839
Epoch 11/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 494ms/step - accuracy: 0.1358 - loss: 2.6145 - val_accuracy: 0.0846
- val_loss: 2.6752
Epoch 12/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 492ms/step - accuracy: 0.1304 - loss: 2.5819 - val_accuracy: 0.1066
- val_loss: 2.6721
Epoch 13/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 495ms/step - accuracy: 0.1380 - loss: 2.5973 - val_accuracy: 0.1103
- val_loss: 2.6702
Epoch 14/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 498ms/step - accuracy: 0.1469 - loss: 2.5946 - val_accuracy: 0.1213
- val_loss: 2.6631
Epoch 15/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 498ms/step - accuracy: 0.1649 - loss: 2.5547 - val_accuracy: 0.1250
- val_loss: 2.6572
Epoch 16/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 492ms/step - accuracy: 0.1628 - loss: 2.5543 - val_accuracy: 0.1324
- val_loss: 2.6463
Epoch 17/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 494ms/step - accuracy: 0.1764 - loss: 2.5285 - val_accuracy: 0.1691
- val_loss: 2.6319
Epoch 18/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 486ms/step - accuracy: 0.1934 - loss: 2.4937 - val_accuracy: 0.1838
- val_loss: 2.6329
Epoch 19/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 489ms/step - accuracy: 0.1788 - loss: 2.5321 - val_accuracy: 0.1654
- val_loss: 2.6193
Epoch 20/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 488ms/step - accuracy: 0.1824 - loss: 2.5093 - val_accuracy: 0.1728
```

```
- val_loss: 2.6029
Epoch 21/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 491ms/step - accuracy: 0.2022 - loss: 2.4294 - val_accuracy: 0.1728
- val_loss: 2.5817
Epoch 22/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 492ms/step - accuracy: 0.1951 - loss: 2.4603 - val_accuracy: 0.1765
- val_loss: 2.5711
Epoch 23/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 491ms/step - accuracy: 0.2062 - loss: 2.4014 - val_accuracy: 0.1691
- val_loss: 2.5738
Epoch 24/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 492ms/step - accuracy: 0.2104 - loss: 2.4048 - val_accuracy: 0.1875
- val_loss: 2.5501
Epoch 25/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 492ms/step - accuracy: 0.2154 - loss: 2.3847 - val_accuracy: 0.2059
- val_loss: 2.5382
Epoch 26/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 489ms/step - accuracy: 0.2118 - loss: 2.3693 - val_accuracy: 0.2059
- val_loss: 2.5327
Epoch 27/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 493ms/step - accuracy: 0.1954 - loss: 2.3654 - val_accuracy: 0.2096
- val_loss: 2.5258
Epoch 28/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 486ms/step - accuracy: 0.2248 - loss: 2.3146 - val_accuracy: 0.2132
- val_loss: 2.5039
Epoch 29/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 485ms/step - accuracy: 0.2299 - loss: 2.3148 - val_accuracy: 0.2169
- val_loss: 2.4976
Epoch 30/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 488ms/step - accuracy: 0.2244 - loss: 2.3212 - val_accuracy: 0.2059
- val_loss: 2.4853
Epoch 31/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 487ms/step - accuracy: 0.2452 - loss: 2.3163 - val_accuracy: 0.2132
- val_loss: 2.4816
Epoch 32/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 488ms/step - accuracy: 0.2396 - loss: 2.3037 - val_accuracy: 0.2279
- val_loss: 2.4758
Epoch 33/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 483ms/step - accuracy: 0.2521 - loss: 2.2300 - val_accuracy: 0.2279
- val_loss: 2.4613
Epoch 34/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 486ms/step - accuracy: 0.2406 - loss: 2.2471 - val_accuracy: 0.2463
- val_loss: 2.4587
Epoch 35/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 489ms/step - accuracy: 0.2371 - loss: 2.2294 - val_accuracy: 0.2390
- val_loss: 2.4412
Epoch 36/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 492ms/step - accuracy: 0.2567 - loss: 2.2143 - val_accuracy: 0.2426
- val_loss: 2.4335
Epoch 37/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 497ms/step - accuracy: 0.2837 - loss: 2.1290 - val_accuracy: 0.2316
- val_loss: 2.4368
Epoch 38/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 496ms/step - accuracy: 0.2640 - loss: 2.1693 - val_accuracy: 0.2279
- val_loss: 2.4129
Epoch 39/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 496ms/step - accuracy: 0.2460 - loss: 2.2081 - val_accuracy: 0.2353
- val_loss: 2.4045
Epoch 40/50
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 497ms/step - accuracy: 0.2637 - loss: 2.1836 - val_accuracy: 0.2500
- val_loss: 2.3736
Epoch 41/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 492ms/step - accuracy: 0.2825 - loss: 2.0896 - val_accuracy: 0.2610
- val_loss: 2.3472
Epoch 42/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 488ms/step - accuracy: 0.2735 - loss: 2.1077 - val_accuracy: 0.2757
- val_loss: 2.3338
Epoch 43/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 495ms/step - accuracy: 0.3010 - loss: 2.0757 - val_accuracy: 0.2721
- val_loss: 2.3181
Epoch 44/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 488ms/step - accuracy: 0.3151 - loss: 2.0489 - val_accuracy: 0.2831
- val_loss: 2.3186
Epoch 45/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 489ms/step - accuracy: 0.3082 - loss: 2.0445 - val_accuracy: 0.2574
- val_loss: 2.3162
Epoch 46/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 497ms/step - accuracy: 0.3185 - loss: 1.9773 - val_accuracy: 0.2610
- val_loss: 2.3170
Epoch 47/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 494ms/step - accuracy: 0.3350 - loss: 2.0456 - val_accuracy: 0.2537
- val_loss: 2.3444
Epoch 48/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 488ms/step - accuracy: 0.3263 - loss: 1.9783 - val_accuracy: 0.2647
- val_loss: 2.3535
Epoch 49/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 494ms/step - accuracy: 0.3300 - loss: 1.9819 - val_accuracy: 0.2684
- val_loss: 2.3070
Epoch 50/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 502ms/step - accuracy: 0.3534 - loss: 1.9424 - val_accuracy: 0.2574
- val_loss: 2.2980
```

In [25]:

```python
#check metrics on test data and plot accuracy curve

anima_acc_1 = history_CNN_1.history['accuracy']

epochs_ind = [i for i in range(1, 1 + epochs)]

plt.plot(epochs_ind, anima_acc_1, 'blue', label = 'Training accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy over training epochs')
plt.legend()
plt.show()
```



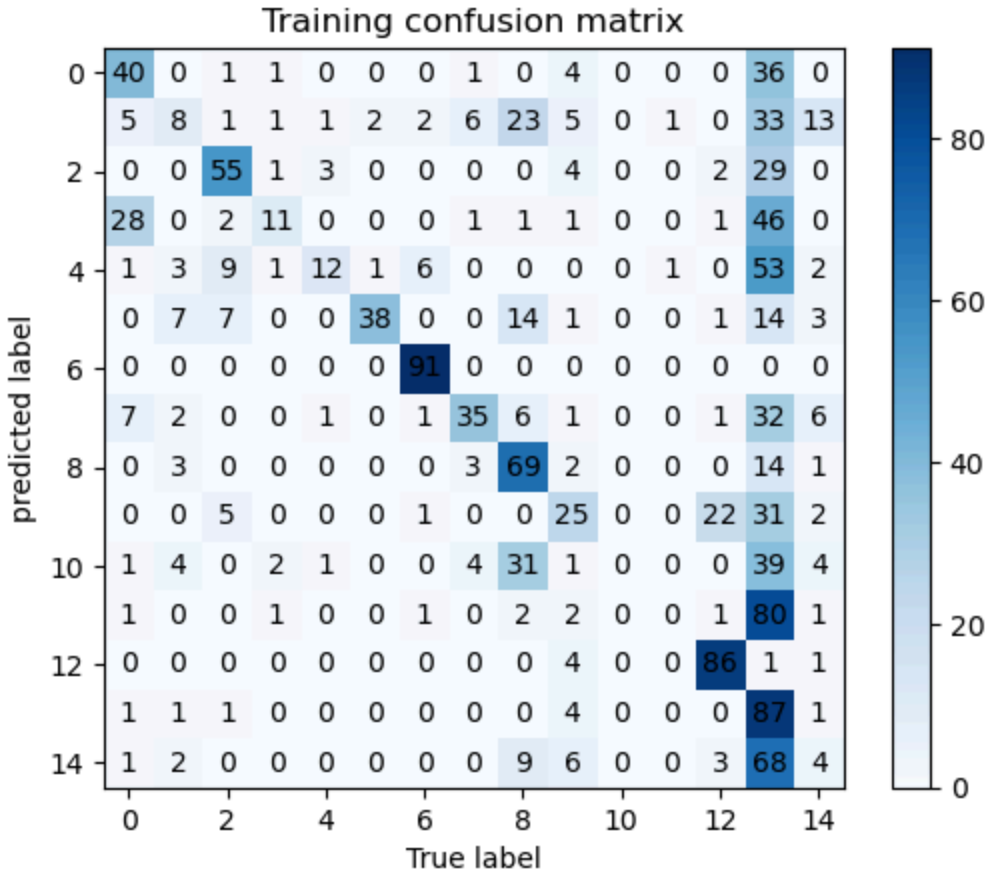
In [26]:
```
_, train_acc = animal_CNN_1.evaluate(X_train, Y_train, verbose = 0)
_, test_acc = animal_CNN_1.evaluate(X_test, Y_test, verbose = 0)

print(f"Training accuracy: {train_acc*100}%")
print(f"Testing accuracy: {test_acc*100}%")
```

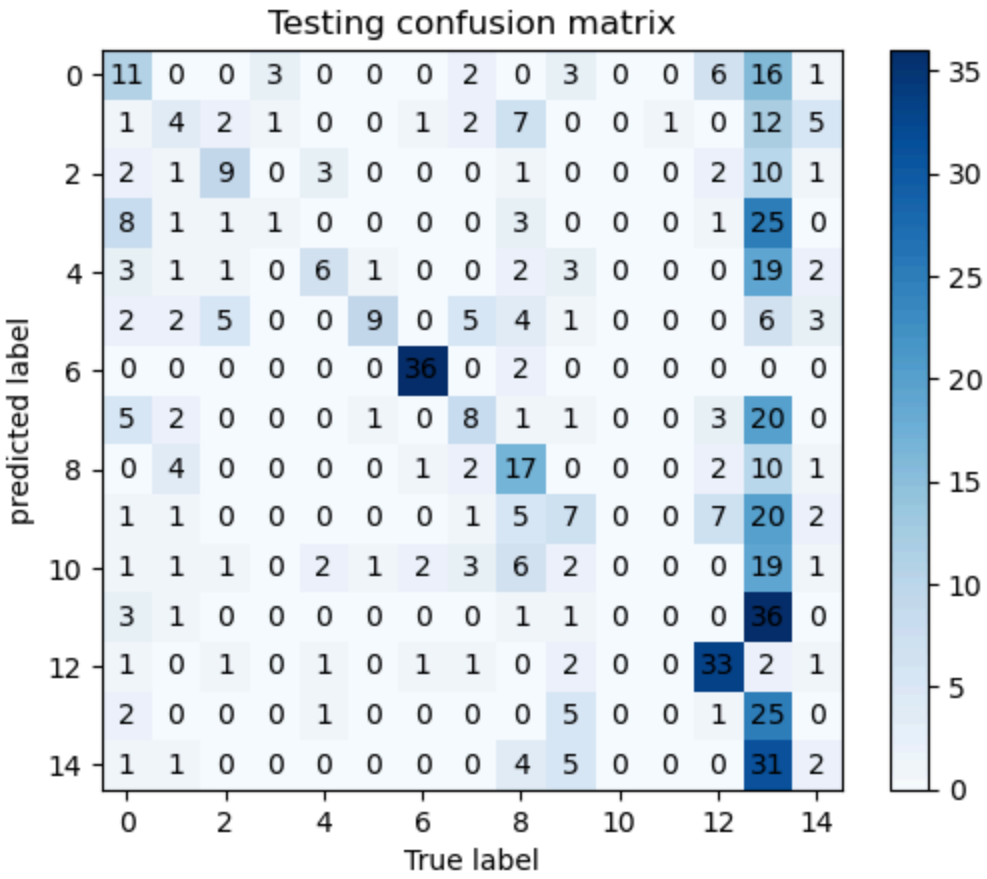
```
Training accuracy: 41.24999940395355%
Testing accuracy: 28.767123818397522%
```

In [27]:
```
#confusion matrix

train_preds = np.argmax(animal_CNN_1.predict(X_train), axis = 1)
test_preds = np.argmax(animal_CNN_1.predict(X_test), axis = 1)

train_cm = confusion_matrix(np.argmax(Y_train, axis = 1), train_preds)
test_cm = confusion_matrix(np.argmax(Y_test, axis = 1), test_preds)

plt.imshow(train_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(train_cm.shape[0]):
  for j in range(train_cm.shape[1]):
    plt.text(j, i, train_cm[i, j], ha = 'center', va = 'center')
plt.title('Training confusion matrix')
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()

plt.imshow(test_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(test_cm.shape[0]):
  for j in range(test_cm.shape[1]):
    plt.text(j, i, test_cm[i, j], ha = 'center', va = 'center')
```

```
plt.title('Testing confusion matrix')
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step
```





This model has significant overfitting present, and additionally took a very long time to complete fitting. I will try a new model architecture which should reduce total parameters and account for some of the overfitting present. More training would liklely result in better performance on train data, but the test data is already far behind in accuracy.

In [12]:
```
from keras.layers import GlobalAveragePooling2D
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
from keras.layers import BatchNormalization, Activation
```

In [13]:
```
animal_CNN_2 = Sequential()
animal_CNN_2.add(InputLayer(input_shape = X_shape[1:]))
animal_CNN_2.add(Conv2D(filters = 32,
                        kernel_size = (3, 3),
                        activation = None,
                        padding = 'same',
                        kernel_regularizer = regularizers.l2(0.001)))
animal_CNN_2.add(BatchNormalization())
animal_CNN_2.add(Activation('relu'))
animal_CNN_2.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_2.add(Dropout(0.25))

animal_CNN_2.add(Conv2D(filters = 32,
                        kernel_size = (3, 3),
                        padding = 'same',
                        activation = None,
                        kernel_regularizer = regularizers.l2(0.001)))
animal_CNN_2.add(BatchNormalization())
animal_CNN_2.add(Activation('relu'))
animal_CNN_2.add(MaxPooling2D(pool_size = (2,2)))
```

```
animal_CNN_2.add(Dropout(0.25))

animal_CNN_2.add(GlobalAveragePooling2D())
animal_CNN_2.add(Dense(128,
                       activation = 'relu',
                       kernel_regularizer = regularizers.l2(0.001)))
animal_CNN_2.add(Dropout(0.25))

animal_CNN_2.add(Dense(Y_shape[1], activation = 'softmax'))

optimizer = Adam(learning_rate = 0.005, amsgrad = True)

animal_CNN_2.compile(loss = 'categorical_crossentropy',
                     optimizer = optimizer,
                     metrics = ['accuracy'])

animal_CNN_2.summary()
```

```
C:\Users\lcleymaet\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\core\input_lay
er.py:27: UserWarning: Argument `input_shape` is deprecated. Use `shape` instead.
  warnings.warn(
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
batch_normalization (BatchNormalization)	(None, 224, 224, 32)	128
activation (Activation)	(None, 224, 224, 32)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
dropout (Dropout)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 112, 112, 32)	128
activation_1 (Activation)	(None, 112, 112, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 32)	0
dropout_1 (Dropout)	(None, 56, 56, 32)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 32)	0
dense (Dense)	(None, 128)	4,224
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 15)	1,935

Total params: 16,559 (64.68 KB)

Trainable params: 16,431 (64.18 KB)

Non-trainable params: 128 (512.00 B)

This clearly has significantly fewer parameters so should reduce the overfitting we saw in the earlier iteration. I will conduct this with more epochs as it should also run much faster.

In [14]:
```
#hyper parameters
epochs = 150
batch_size = 128
early_stop = EarlyStopping(monitor = 'val_loss', patience = 5, restore_best_weights = True)
history_CNN_2 = animal_CNN_2.fit(X_train,
                                 Y_train,
                                 epochs = epochs,
                                 batch_size = batch_size,
                                validation_split = 0.2)
```

```
Epoch 1/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 22s 1s/step - accuracy: 0.1042 - loss: 2.7988 - val_accuracy: 0.1213 - v
al_loss: 2.7845
Epoch 2/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.1511 - loss: 2.6652 - val_accuracy: 0.0919 - v
al_loss: 2.7767
Epoch 3/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.1815 - loss: 2.5803 - val_accuracy: 0.1140 - v
al_loss: 2.7505
Epoch 4/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.1976 - loss: 2.5222 - val_accuracy: 0.1140 - v
al_loss: 2.7441
Epoch 5/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.2161 - loss: 2.4852 - val_accuracy: 0.1250 - v
al_loss: 2.7598
Epoch 6/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.2241 - loss: 2.4355 - val_accuracy: 0.1140 - v
al_loss: 2.7635
Epoch 7/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.2693 - loss: 2.3680 - val_accuracy: 0.1029 - v
al_loss: 2.8044
Epoch 8/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.2553 - loss: 2.3567 - val_accuracy: 0.1066 - v
al_loss: 2.8899
Epoch 9/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.2758 - loss: 2.3107 - val_accuracy: 0.1140 - v
al_loss: 2.9764
Epoch 10/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.2825 - loss: 2.2824 - val_accuracy: 0.1066 - v
al_loss: 3.1117
Epoch 11/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.2911 - loss: 2.2607 - val_accuracy: 0.1103 - v
al_loss: 3.3496
Epoch 12/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3077 - loss: 2.2211 - val_accuracy: 0.1103 - v
al_loss: 3.3147
Epoch 13/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3062 - loss: 2.2102 - val_accuracy: 0.0956 - v
al_loss: 3.4226
Epoch 14/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3071 - loss: 2.1576 - val_accuracy: 0.1066 - v
al_loss: 3.8489
Epoch 15/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3324 - loss: 2.1229 - val_accuracy: 0.1066 - v
al_loss: 4.0477
Epoch 16/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3184 - loss: 2.1340 - val_accuracy: 0.1029 - v
al_loss: 4.3036
Epoch 17/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3175 - loss: 2.1428 - val_accuracy: 0.1066 - v
al_loss: 4.4169
Epoch 18/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3307 - loss: 2.1123 - val_accuracy: 0.1066 - v
al_loss: 4.5832
Epoch 19/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3442 - loss: 2.1070 - val_accuracy: 0.1066 - v
al_loss: 4.3708
Epoch 20/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3195 - loss: 2.1121 - val_accuracy: 0.1360 - v
```

```
al_loss: 4.3159
Epoch 21/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3414 - loss: 2.0523 - val_accuracy: 0.1140 - v
al_loss: 4.8894
Epoch 22/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3552 - loss: 2.0587 - val_accuracy: 0.1103 - v
al_loss: 4.6397
Epoch 23/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3536 - loss: 2.0336 - val_accuracy: 0.1471 - v
al_loss: 4.0938
Epoch 24/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3390 - loss: 2.0500 - val_accuracy: 0.1434 - v
al_loss: 4.2552
Epoch 25/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3808 - loss: 1.9657 - val_accuracy: 0.1434 - v
al_loss: 4.6587
Epoch 26/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3542 - loss: 2.0085 - val_accuracy: 0.1360 - v
al_loss: 4.3631
Epoch 27/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3880 - loss: 1.9561 - val_accuracy: 0.1213 - v
al_loss: 4.7218
Epoch 28/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3779 - loss: 1.9276 - val_accuracy: 0.1140 - v
al_loss: 5.1410
Epoch 29/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3774 - loss: 1.9317 - val_accuracy: 0.1581 - v
al_loss: 4.7118
Epoch 30/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3903 - loss: 1.9306 - val_accuracy: 0.1287 - v
al_loss: 5.2924
Epoch 31/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4061 - loss: 1.9229 - val_accuracy: 0.1360 - v
al_loss: 5.6270
Epoch 32/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3924 - loss: 1.9241 - val_accuracy: 0.1434 - v
al_loss: 4.8616
Epoch 33/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3824 - loss: 1.9227 - val_accuracy: 0.1471 - v
al_loss: 4.5534
Epoch 34/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4240 - loss: 1.8612 - val_accuracy: 0.1397 - v
al_loss: 4.9313
Epoch 35/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4109 - loss: 1.8490 - val_accuracy: 0.1507 - v
al_loss: 5.1306
Epoch 36/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4136 - loss: 1.8591 - val_accuracy: 0.1507 - v
al_loss: 5.5890
Epoch 37/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4038 - loss: 1.8518 - val_accuracy: 0.1140 - v
al_loss: 6.7379
Epoch 38/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4186 - loss: 1.8486 - val_accuracy: 0.1507 - v
al_loss: 4.8283
Epoch 39/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4207 - loss: 1.8527 - val_accuracy: 0.1434 - v
al_loss: 5.7209
Epoch 40/150
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4323 - loss: 1.8288 - val_accuracy: 0.1360 - v
al_loss: 5.6326
Epoch 41/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4164 - loss: 1.8444 - val_accuracy: 0.1507 - v
al_loss: 4.7133
Epoch 42/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4584 - loss: 1.7231 - val_accuracy: 0.1581 - v
al_loss: 5.5010
Epoch 43/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4759 - loss: 1.7098 - val_accuracy: 0.1618 - v
al_loss: 4.5630
Epoch 44/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4577 - loss: 1.7305 - val_accuracy: 0.1838 - v
al_loss: 4.2276
Epoch 45/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4629 - loss: 1.6981 - val_accuracy: 0.1875 - v
al_loss: 3.9867
Epoch 46/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.4616 - loss: 1.7340 - val_accuracy: 0.2096 - v
al_loss: 3.8694
Epoch 47/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4508 - loss: 1.7849 - val_accuracy: 0.1912 - v
al_loss: 3.7607
Epoch 48/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4873 - loss: 1.7105 - val_accuracy: 0.1507 - v
al_loss: 3.6803
Epoch 49/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4779 - loss: 1.6997 - val_accuracy: 0.2794 - v
al_loss: 2.8196
Epoch 50/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4690 - loss: 1.7303 - val_accuracy: 0.2868 - v
al_loss: 2.5787
Epoch 51/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4866 - loss: 1.6819 - val_accuracy: 0.2132 - v
al_loss: 2.8849
Epoch 52/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4794 - loss: 1.6880 - val_accuracy: 0.2684 - v
al_loss: 2.6112
Epoch 53/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4977 - loss: 1.6695 - val_accuracy: 0.2757 - v
al_loss: 3.0820
Epoch 54/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4604 - loss: 1.7344 - val_accuracy: 0.2978 - v
al_loss: 2.7038
Epoch 55/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4900 - loss: 1.6398 - val_accuracy: 0.2537 - v
al_loss: 2.7447
Epoch 56/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4928 - loss: 1.6764 - val_accuracy: 0.2941 - v
al_loss: 2.6568
Epoch 57/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4956 - loss: 1.6331 - val_accuracy: 0.2206 - v
al_loss: 3.6687
Epoch 58/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5058 - loss: 1.6375 - val_accuracy: 0.2426 - v
al_loss: 3.3172
Epoch 59/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4917 - loss: 1.6408 - val_accuracy: 0.2279 - v
al_loss: 3.0617
```

```
Epoch 60/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5352 - loss: 1.6028 - val_accuracy: 0.2610 - v
al_loss: 2.7982
Epoch 61/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5077 - loss: 1.6296 - val_accuracy: 0.2096 - v
al_loss: 3.3927
Epoch 62/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5064 - loss: 1.6450 - val_accuracy: 0.2353 - v
al_loss: 3.2290
Epoch 63/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4959 - loss: 1.6752 - val_accuracy: 0.2831 - v
al_loss: 2.6911
Epoch 64/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.4639 - loss: 1.7074 - val_accuracy: 0.2169 - v
al_loss: 3.5022
Epoch 65/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5070 - loss: 1.6461 - val_accuracy: 0.2279 - v
al_loss: 2.9428
Epoch 66/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5277 - loss: 1.6363 - val_accuracy: 0.2463 - v
al_loss: 2.8553
Epoch 67/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5023 - loss: 1.6269 - val_accuracy: 0.2757 - v
al_loss: 2.8124
Epoch 68/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5078 - loss: 1.6057 - val_accuracy: 0.2941 - v
al_loss: 2.5603
Epoch 69/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.5348 - loss: 1.5645 - val_accuracy: 0.2426 - v
al_loss: 2.9081
Epoch 70/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5287 - loss: 1.5825 - val_accuracy: 0.3199 - v
al_loss: 2.6018
Epoch 71/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5248 - loss: 1.6310 - val_accuracy: 0.3199 - v
al_loss: 2.2742
Epoch 72/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5512 - loss: 1.5332 - val_accuracy: 0.3713 - v
al_loss: 2.2615
Epoch 73/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5202 - loss: 1.5858 - val_accuracy: 0.3676 - v
al_loss: 2.4460
Epoch 74/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5352 - loss: 1.5550 - val_accuracy: 0.3309 - v
al_loss: 2.2591
Epoch 75/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5449 - loss: 1.5412 - val_accuracy: 0.2831 - v
al_loss: 2.5679
Epoch 76/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5262 - loss: 1.5610 - val_accuracy: 0.2390 - v
al_loss: 3.0202
Epoch 77/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5284 - loss: 1.5890 - val_accuracy: 0.1544 - v
al_loss: 3.6163
Epoch 78/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5465 - loss: 1.5380 - val_accuracy: 0.1765 - v
al_loss: 3.7484
Epoch 79/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5614 - loss: 1.5149 - val_accuracy: 0.2610 - v
```

```
al_loss: 2.8394
Epoch 80/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5570 - loss: 1.5006 - val_accuracy: 0.1985 - v
al_loss: 3.1030
Epoch 81/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5629 - loss: 1.4994 - val_accuracy: 0.1507 - v
al_loss: 3.7986
Epoch 82/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5560 - loss: 1.5511 - val_accuracy: 0.2243 - v
al_loss: 3.0279
Epoch 83/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.5588 - loss: 1.5252 - val_accuracy: 0.2426 - v
al_loss: 3.0617
Epoch 84/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5428 - loss: 1.5525 - val_accuracy: 0.2316 - v
al_loss: 2.7968
Epoch 85/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5548 - loss: 1.5133 - val_accuracy: 0.1507 - v
al_loss: 4.2546
Epoch 86/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5764 - loss: 1.4807 - val_accuracy: 0.2022 - v
al_loss: 3.2575
Epoch 87/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5758 - loss: 1.5005 - val_accuracy: 0.2574 - v
al_loss: 2.9357
Epoch 88/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5517 - loss: 1.4753 - val_accuracy: 0.2390 - v
al_loss: 3.0708
Epoch 89/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5837 - loss: 1.4754 - val_accuracy: 0.2831 - v
al_loss: 2.8128
Epoch 90/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5796 - loss: 1.4702 - val_accuracy: 0.1949 - v
al_loss: 3.6536
Epoch 91/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5419 - loss: 1.5435 - val_accuracy: 0.3051 - v
al_loss: 2.6647
Epoch 92/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5548 - loss: 1.5035 - val_accuracy: 0.1912 - v
al_loss: 3.5857
Epoch 93/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5802 - loss: 1.4500 - val_accuracy: 0.2059 - v
al_loss: 3.4239
Epoch 94/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5405 - loss: 1.5156 - val_accuracy: 0.2904 - v
al_loss: 2.6082
Epoch 95/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6030 - loss: 1.4019 - val_accuracy: 0.2537 - v
al_loss: 3.2535
Epoch 96/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5611 - loss: 1.4919 - val_accuracy: 0.2868 - v
al_loss: 3.0926
Epoch 97/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5587 - loss: 1.4786 - val_accuracy: 0.2537 - v
al_loss: 2.9935
Epoch 98/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5551 - loss: 1.4611 - val_accuracy: 0.3382 - v
al_loss: 2.7183
Epoch 99/150
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5738 - loss: 1.4769 - val_accuracy: 0.3272 - v
al_loss: 2.8975
Epoch 100/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5867 - loss: 1.4897 - val_accuracy: 0.2904 - v
al_loss: 2.8722
Epoch 101/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5790 - loss: 1.4364 - val_accuracy: 0.3529 - v
al_loss: 2.6261
Epoch 102/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5793 - loss: 1.4248 - val_accuracy: 0.3824 - v
al_loss: 2.4304
Epoch 103/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5926 - loss: 1.4643 - val_accuracy: 0.2390 - v
al_loss: 3.5672
Epoch 104/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5682 - loss: 1.4860 - val_accuracy: 0.3088 - v
al_loss: 2.6408
Epoch 105/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5834 - loss: 1.4481 - val_accuracy: 0.3235 - v
al_loss: 2.5232
Epoch 106/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5616 - loss: 1.4855 - val_accuracy: 0.2868 - v
al_loss: 3.1407
Epoch 107/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5684 - loss: 1.4620 - val_accuracy: 0.3382 - v
al_loss: 2.5804
Epoch 108/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5623 - loss: 1.5494 - val_accuracy: 0.3456 - v
al_loss: 2.7133
Epoch 109/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6074 - loss: 1.4100 - val_accuracy: 0.3787 - v
al_loss: 2.4475
Epoch 110/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6098 - loss: 1.4050 - val_accuracy: 0.3125 - v
al_loss: 2.9862
Epoch 111/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5968 - loss: 1.4260 - val_accuracy: 0.3824 - v
al_loss: 2.4999
Epoch 112/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6019 - loss: 1.4210 - val_accuracy: 0.3199 - v
al_loss: 2.5447
Epoch 113/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6270 - loss: 1.3388 - val_accuracy: 0.2721 - v
al_loss: 3.1984
Epoch 114/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6141 - loss: 1.3407 - val_accuracy: 0.3235 - v
al_loss: 3.0374
Epoch 115/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5921 - loss: 1.4533 - val_accuracy: 0.3419 - v
al_loss: 2.5163
Epoch 116/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6142 - loss: 1.3662 - val_accuracy: 0.3860 - v
al_loss: 2.2556
Epoch 117/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5974 - loss: 1.3637 - val_accuracy: 0.4301 - v
al_loss: 2.2553
Epoch 118/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6067 - loss: 1.3672 - val_accuracy: 0.3934 - v
al_loss: 2.3601
```

```
Epoch 119/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6265 - loss: 1.3653 - val_accuracy: 0.4044 - v
al_loss: 2.3183
Epoch 120/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6401 - loss: 1.3328 - val_accuracy: 0.3787 - v
al_loss: 2.3525
Epoch 121/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6342 - loss: 1.3140 - val_accuracy: 0.4044 - v
al_loss: 2.3309
Epoch 122/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6318 - loss: 1.2991 - val_accuracy: 0.3419 - v
al_loss: 2.5404
Epoch 123/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6351 - loss: 1.3313 - val_accuracy: 0.3272 - v
al_loss: 2.6484
Epoch 124/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5633 - loss: 1.4821 - val_accuracy: 0.3897 - v
al_loss: 2.4525
Epoch 125/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5943 - loss: 1.3963 - val_accuracy: 0.3640 - v
al_loss: 2.5286
Epoch 126/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5968 - loss: 1.3876 - val_accuracy: 0.3603 - v
al_loss: 2.6499
Epoch 127/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6010 - loss: 1.3760 - val_accuracy: 0.4412 - v
al_loss: 2.2170
Epoch 128/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5979 - loss: 1.3577 - val_accuracy: 0.2831 - v
al_loss: 3.0691
Epoch 129/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6297 - loss: 1.3407 - val_accuracy: 0.3015 - v
al_loss: 2.8644
Epoch 130/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6199 - loss: 1.2944 - val_accuracy: 0.4191 - v
al_loss: 2.2702
Epoch 131/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6471 - loss: 1.3079 - val_accuracy: 0.3640 - v
al_loss: 2.5465
Epoch 132/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6130 - loss: 1.3927 - val_accuracy: 0.3640 - v
al_loss: 2.5889
Epoch 133/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6179 - loss: 1.3438 - val_accuracy: 0.3456 - v
al_loss: 2.6738
Epoch 134/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6356 - loss: 1.3279 - val_accuracy: 0.2868 - v
al_loss: 3.1170
Epoch 135/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6225 - loss: 1.3564 - val_accuracy: 0.3051 - v
al_loss: 2.7538
Epoch 136/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6572 - loss: 1.2998 - val_accuracy: 0.2647 - v
al_loss: 3.8451
Epoch 137/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6202 - loss: 1.2826 - val_accuracy: 0.2426 - v
al_loss: 3.4253
Epoch 138/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6250 - loss: 1.3618 - val_accuracy: 0.2978 - v
```

```
al_loss: 3.1878
Epoch 139/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6362 - loss: 1.2732 - val_accuracy: 0.3750 - v
al_loss: 2.4932
Epoch 140/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6403 - loss: 1.3025 - val_accuracy: 0.3713 - v
al_loss: 2.4168
Epoch 141/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6329 - loss: 1.3077 - val_accuracy: 0.4007 - v
al_loss: 2.3334
Epoch 142/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6428 - loss: 1.2895 - val_accuracy: 0.3456 - v
al_loss: 2.7089
Epoch 143/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6473 - loss: 1.2752 - val_accuracy: 0.3676 - v
al_loss: 2.7030
Epoch 144/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6290 - loss: 1.3407 - val_accuracy: 0.3640 - v
al_loss: 2.6306
Epoch 145/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6304 - loss: 1.3109 - val_accuracy: 0.3787 - v
al_loss: 2.3605
Epoch 146/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6198 - loss: 1.3471 - val_accuracy: 0.4375 - v
al_loss: 2.4065
Epoch 147/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6492 - loss: 1.3306 - val_accuracy: 0.4485 - v
al_loss: 2.1531
Epoch 148/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6596 - loss: 1.3001 - val_accuracy: 0.4301 - v
al_loss: 2.2341
Epoch 149/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6493 - loss: 1.2900 - val_accuracy: 0.3419 - v
al_loss: 2.5952
Epoch 150/150
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.6471 - loss: 1.2809 - val_accuracy: 0.3750 - v
al_loss: 2.4743
```

In [15]:

```python
#check metrics on test data and plot accuracy curve

animal_acc_2 = history_CNN_2.history['accuracy']

epochs_ind = [i for i in range(1, 1 + epochs)]

plt.plot(epochs_ind, animal_acc_2, 'blue', label = 'Training accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy over training epochs')
plt.legend()
plt.show()
```

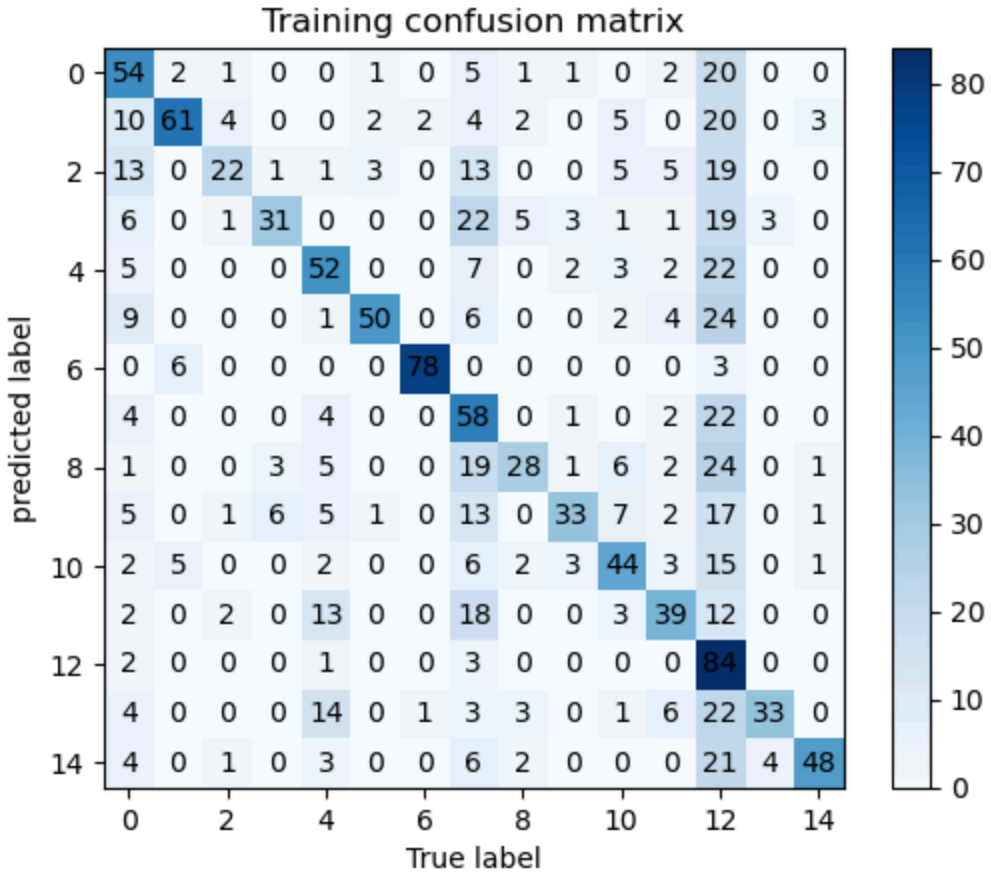
In [16]:
```
_, train_acc = animal_CNN_2.evaluate(X_train, Y_train, verbose = 0)
_, test_acc = animal_CNN_2.evaluate(X_test, Y_test, verbose = 0)

print(f"Training accuracy: {train_acc*100}%")
print(f"Testing accuracy: {test_acc*100}%")
```

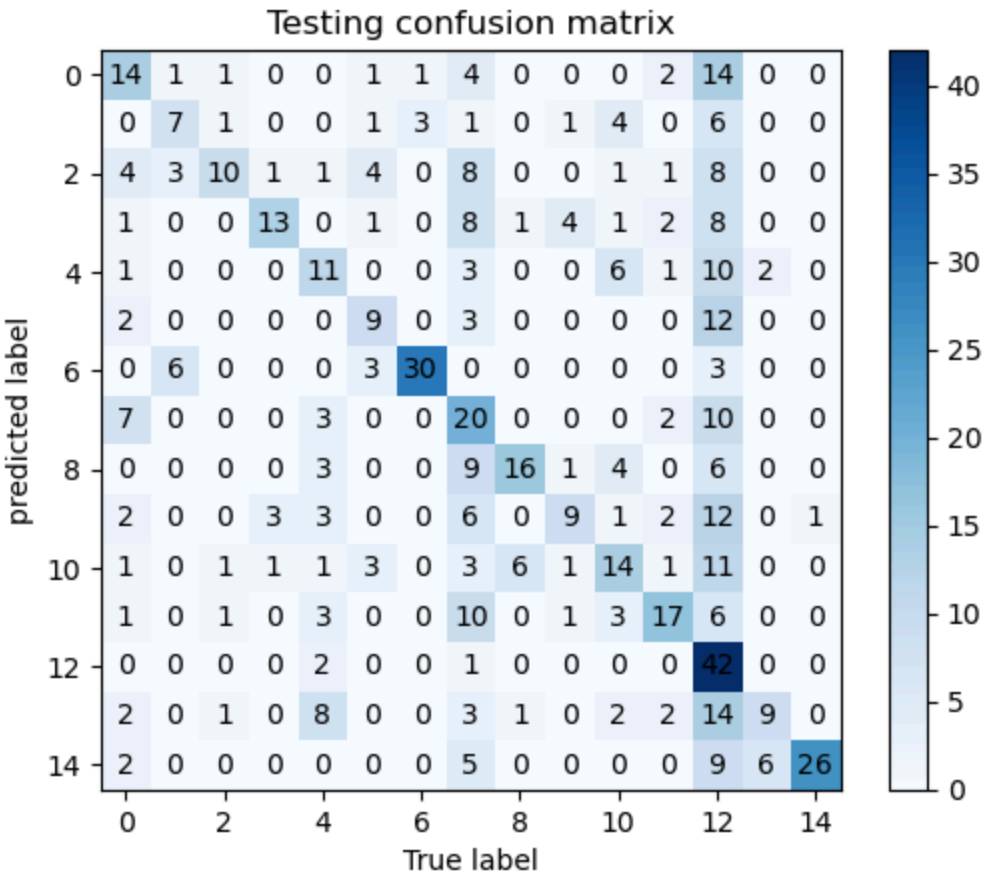
```
Training accuracy: 52.57353186607361%
Testing accuracy: 42.294520139694214%
```

In [17]:
```
#confusion matrix

train_preds = np.argmax(animal_CNN_2.predict(X_train), axis = 1)
test_preds = np.argmax(animal_CNN_2.predict(X_test), axis = 1)

train_cm = confusion_matrix(np.argmax(Y_train, axis = 1), train_preds)
test_cm = confusion_matrix(np.argmax(Y_test, axis = 1), test_preds)

plt.imshow(train_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(train_cm.shape[0]):
  for j in range(train_cm.shape[1]):
    plt.text(j, i, train_cm[i, j], ha = 'center', va = 'center')
plt.title('Training confusion matrix')
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()

plt.imshow(test_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(test_cm.shape[0]):
  for j in range(test_cm.shape[1]):
    plt.text(j, i, test_cm[i, j], ha = 'center', va = 'center')
```

```
plt.title('Testing confusion matrix')
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step
```





While this is improved, the accuracy is not ideal and I still am seeing overfitting. I will now add transformations of the images as well as allow the learning rate to decay over time if a plateau is reached.

In [13]:
```
#add image transformations

from tensorflow.keras.preprocessing.image import ImageDataGenerator

generator = ImageDataGenerator(
    rotation_range = 40,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    horizontal_flip = True,
    vertical_flip = True,
    zoom_range = 0.1,
    validation_split = 0.2
)
```

In [14]:
```
train_gen = generator.flow(X_train, Y_train, subset = "training", batch_size = 500)
val_gen = generator.flow(X_train, Y_train, subset = "validation", batch_size = 500)
```

In [50]:
```
#new model architecture
animal_CNN_3 = Sequential()
animal_CNN_3.add(InputLayer(input_shape = X_shape[1:]))
animal_CNN_3.add(Conv2D(filters = 64,
                        kernel_size = (3, 3),
                        activation = None,
                        padding = 'same',
                        kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_3.add(BatchNormalization())
animal_CNN_3.add(Activation('relu'))
```

```
animal_CNN_3.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_3.add(Dropout(0.25))

animal_CNN_3.add(Conv2D(filters = 64,
                        kernel_size = (3, 3),
                        activation = None,
                        padding = 'same',
                        kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_3.add(BatchNormalization())
animal_CNN_3.add(Activation('relu'))
animal_CNN_3.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_3.add(Dropout(0.25))

animal_CNN_3.add(Conv2D(filters = 32,
                        kernel_size = (3, 3),
                        padding = 'same',
                        activation = None,
                        kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_3.add(BatchNormalization())
animal_CNN_3.add(Activation('relu'))
animal_CNN_3.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_3.add(Dropout(0.25))

animal_CNN_3.add(GlobalAveragePooling2D())
animal_CNN_3.add(Dense(64,
                       activation = 'relu',
                       kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_3.add(Dropout(0.1))

animal_CNN_3.add(Dense(Y_shape[1], activation = 'softmax'))

optimizer = Adam(learning_rate = 0.01, amsgrad = True)

animal_CNN_3.compile(loss = 'categorical_crossentropy',
                     optimizer = optimizer,
                     metrics = ['accuracy'])

animal_CNN_3.summary()
```

Model: "sequential_10"

Layer (type)	Output Shape	Param #
conv2d_26 (Conv2D)	(None, 224, 224, 64)	1,792
batch_normalization_26 (BatchNormalization)	(None, 224, 224, 64)	256
activation_26 (Activation)	(None, 224, 224, 64)	0
max_pooling2d_26 (MaxPooling2D)	(None, 112, 112, 64)	0
dropout_37 (Dropout)	(None, 112, 112, 64)	0
conv2d_27 (Conv2D)	(None, 112, 112, 64)	36,928
batch_normalization_27 (BatchNormalization)	(None, 112, 112, 64)	256
activation_27 (Activation)	(None, 112, 112, 64)	0
max_pooling2d_27 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_38 (Dropout)	(None, 56, 56, 64)	0
conv2d_28 (Conv2D)	(None, 56, 56, 32)	18,464
batch_normalization_28 (BatchNormalization)	(None, 56, 56, 32)	128
activation_28 (Activation)	(None, 56, 56, 32)	0
max_pooling2d_28 (MaxPooling2D)	(None, 28, 28, 32)	0
dropout_39 (Dropout)	(None, 28, 28, 32)	0
global_average_pooling2d_7 (GlobalAveragePooling2D)	(None, 32)	0
dense_21 (Dense)	(None, 64)	2,112
dropout_40 (Dropout)	(None, 64)	0
dense_22 (Dense)	(None, 15)	975

Total params: 60,911 (237.93 KB)

Trainable params: 60,591 (236.68 KB)

Non-trainable params: 320 (1.25 KB)

In [15]:
```
from keras.callbacks import ReduceLROnPlateau
```

In [51]:
```
#reduces learning rate if loss value plateaus for 3 iterations
reduce = ReduceLROnPlateau(
    monitor = 'val_loss',
    patience = 3,
    verbose = 1,
    factor = 0.5,
    min_lr = 0.00005
```

```
)

#train the model
epochs = 150
batch_size = 64

history_CNN_3 = animal_CNN_3.fit(train_gen,
                                 epochs = epochs,
                                 batch_size = batch_size,
                                validation_data = val_gen,
                               callbacks = [reduce])
```

```
Epoch 1/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 51s 13s/step - accuracy: 0.0557 - loss: 3.1416 - val_accuracy: 0.1029 -
val_loss: 3.0892 - learning_rate: 0.0100
Epoch 2/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.0991 - loss: 2.7400 - val_accuracy: 0.1029 -
val_loss: 3.3950 - learning_rate: 0.0100
Epoch 3/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.1186 - loss: 2.7213 - val_accuracy: 0.0809 -
val_loss: 3.8704 - learning_rate: 0.0100
Epoch 4/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.1507 - loss: 2.6932
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.004999999888241291.
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.1491 - loss: 2.6945 - val_accuracy: 0.0919 -
val_loss: 3.9390 - learning_rate: 0.0100
Epoch 5/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1450 - loss: 2.6853 - val_accuracy: 0.0846 -
val_loss: 4.0834 - learning_rate: 0.0050
Epoch 6/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.1681 - loss: 2.6654 - val_accuracy: 0.0846 -
val_loss: 4.0244 - learning_rate: 0.0050
Epoch 7/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.1664 - loss: 2.6467
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0024999999441206455.
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1650 - loss: 2.6489 - val_accuracy: 0.0846 -
val_loss: 3.9542 - learning_rate: 0.0050
Epoch 8/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1492 - loss: 2.6610 - val_accuracy: 0.0919 -
val_loss: 3.6172 - learning_rate: 0.0025
Epoch 9/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1524 - loss: 2.6538 - val_accuracy: 0.0956 -
val_loss: 3.2202 - learning_rate: 0.0025
Epoch 10/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.1610 - loss: 2.6285 - val_accuracy: 0.1140 -
val_loss: 2.9673 - learning_rate: 0.0025
Epoch 11/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1547 - loss: 2.6136 - val_accuracy: 0.1176 -
val_loss: 2.8641 - learning_rate: 0.0025
Epoch 12/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1475 - loss: 2.6252 - val_accuracy: 0.1103 -
val_loss: 2.7977 - learning_rate: 0.0025
Epoch 13/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1591 - loss: 2.5963 - val_accuracy: 0.1250 -
val_loss: 2.7976 - learning_rate: 0.0025
Epoch 14/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1497 - loss: 2.6012 - val_accuracy: 0.1140 -
val_loss: 2.7908 - learning_rate: 0.0025
Epoch 15/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1791 - loss: 2.6036 - val_accuracy: 0.1103 -
val_loss: 2.7856 - learning_rate: 0.0025
Epoch 16/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1990 - loss: 2.5496 - val_accuracy: 0.1360 -
val_loss: 2.7765 - learning_rate: 0.0025
Epoch 17/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1873 - loss: 2.5460 - val_accuracy: 0.1434 -
val_loss: 2.7482 - learning_rate: 0.0025
Epoch 18/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 17s/step - accuracy: 0.2169 - loss: 2.5333 - val_accuracy: 0.1507 -
val_loss: 2.7368 - learning_rate: 0.0025
Epoch 19/150
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2033 - loss: 2.5251 - val_accuracy: 0.1213 -
val_loss: 2.7248 - learning_rate: 0.0025
Epoch 20/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2179 - loss: 2.5062 - val_accuracy: 0.1287 -
val_loss: 2.7035 - learning_rate: 0.0025
Epoch 21/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2212 - loss: 2.4845 - val_accuracy: 0.1176 -
val_loss: 2.6997 - learning_rate: 0.0025
Epoch 22/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2272 - loss: 2.4928 - val_accuracy: 0.1066 -
val_loss: 2.7228 - learning_rate: 0.0025
Epoch 23/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2527 - loss: 2.4305 - val_accuracy: 0.0809 -
val_loss: 2.7826 - learning_rate: 0.0025
Epoch 24/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.2308 - loss: 2.4525
Epoch 24: ReduceLROnPlateau reducing learning rate to 0.0012499999720603228.
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2303 - loss: 2.4522 - val_accuracy: 0.0956 -
val_loss: 2.7569 - learning_rate: 0.0025
Epoch 25/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2303 - loss: 2.4367 - val_accuracy: 0.0956 -
val_loss: 2.7348 - learning_rate: 0.0012
Epoch 26/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2200 - loss: 2.4160 - val_accuracy: 0.1066 -
val_loss: 2.7234 - learning_rate: 0.0012
Epoch 27/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8s/step - accuracy: 0.2367 - loss: 2.4288
Epoch 27: ReduceLROnPlateau reducing learning rate to 0.0006249999860301614.
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2394 - loss: 2.4252 - val_accuracy: 0.0882 -
val_loss: 2.7477 - learning_rate: 0.0012
Epoch 28/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2299 - loss: 2.4035 - val_accuracy: 0.1029 -
val_loss: 2.7416 - learning_rate: 6.2500e-04
Epoch 29/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2391 - loss: 2.3968 - val_accuracy: 0.1066 -
val_loss: 2.7440 - learning_rate: 6.2500e-04
Epoch 30/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.2345 - loss: 2.3988
Epoch 30: ReduceLROnPlateau reducing learning rate to 0.0003124999930150807.
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2340 - loss: 2.3992 - val_accuracy: 0.1103 -
val_loss: 2.7577 - learning_rate: 6.2500e-04
Epoch 31/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2419 - loss: 2.3849 - val_accuracy: 0.0993 -
val_loss: 2.7660 - learning_rate: 3.1250e-04
Epoch 32/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2457 - loss: 2.3816 - val_accuracy: 0.0956 -
val_loss: 2.7665 - learning_rate: 3.1250e-04
Epoch 33/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 15s/step - accuracy: 0.2523 - loss: 2.3739
Epoch 33: ReduceLROnPlateau reducing learning rate to 0.00015624999650754035.
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2534 - loss: 2.3697 - val_accuracy: 0.0919 -
val_loss: 2.7591 - learning_rate: 3.1250e-04
Epoch 34/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2225 - loss: 2.4080 - val_accuracy: 0.0993 -
val_loss: 2.7462 - learning_rate: 1.5625e-04
Epoch 35/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2359 - loss: 2.3690 - val_accuracy: 0.1029 -
val_loss: 2.7415 - learning_rate: 1.5625e-04
Epoch 36/150
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.2468 - loss: 2.3632
Epoch 36: ReduceLROnPlateau reducing learning rate to 7.812499825377017e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2485 - loss: 2.3632 - val_accuracy: 0.1066 -
val_loss: 2.7380 - learning_rate: 1.5625e-04
Epoch 37/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2371 - loss: 2.3746 - val_accuracy: 0.1287 -
val_loss: 2.7348 - learning_rate: 7.8125e-05
Epoch 38/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2768 - loss: 2.3581 - val_accuracy: 0.1213 -
val_loss: 2.7329 - learning_rate: 7.8125e-05
Epoch 39/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.2542 - loss: 2.3508
Epoch 39: ReduceLROnPlateau reducing learning rate to 5e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2554 - loss: 2.3502 - val_accuracy: 0.1250 -
val_loss: 2.7317 - learning_rate: 7.8125e-05
Epoch 40/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2549 - loss: 2.3370 - val_accuracy: 0.1397 -
val_loss: 2.7243 - learning_rate: 5.0000e-05
Epoch 41/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2511 - loss: 2.3433 - val_accuracy: 0.1360 -
val_loss: 2.7186 - learning_rate: 5.0000e-05
Epoch 42/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2483 - loss: 2.3592 - val_accuracy: 0.1434 -
val_loss: 2.7159 - learning_rate: 5.0000e-05
Epoch 43/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2505 - loss: 2.3587 - val_accuracy: 0.1507 -
val_loss: 2.7124 - learning_rate: 5.0000e-05
Epoch 44/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2481 - loss: 2.3720 - val_accuracy: 0.1434 -
val_loss: 2.7102 - learning_rate: 5.0000e-05
Epoch 45/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2316 - loss: 2.3628 - val_accuracy: 0.1397 -
val_loss: 2.7058 - learning_rate: 5.0000e-05
Epoch 46/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2542 - loss: 2.3412 - val_accuracy: 0.1360 -
val_loss: 2.6995 - learning_rate: 5.0000e-05
Epoch 47/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2541 - loss: 2.3515 - val_accuracy: 0.1471 -
val_loss: 2.6938 - learning_rate: 5.0000e-05
Epoch 48/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2505 - loss: 2.3542 - val_accuracy: 0.1507 -
val_loss: 2.6895 - learning_rate: 5.0000e-05
Epoch 49/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2327 - loss: 2.3628 - val_accuracy: 0.1434 -
val_loss: 2.6864 - learning_rate: 5.0000e-05
Epoch 50/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2605 - loss: 2.3474 - val_accuracy: 0.1544 -
val_loss: 2.6836 - learning_rate: 5.0000e-05
Epoch 51/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2566 - loss: 2.3638 - val_accuracy: 0.1434 -
val_loss: 2.6804 - learning_rate: 5.0000e-05
Epoch 52/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2652 - loss: 2.3549 - val_accuracy: 0.1507 -
val_loss: 2.6741 - learning_rate: 5.0000e-05
Epoch 53/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2773 - loss: 2.3218 - val_accuracy: 0.1544 -
val_loss: 2.6725 - learning_rate: 5.0000e-05
Epoch 54/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2491 - loss: 2.3711 - val_accuracy: 0.1544 -
```

```
val_loss: 2.6731 - learning_rate: 5.0000e-05
Epoch 55/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2489 - loss: 2.3462 - val_accuracy: 0.1618 -
val_loss: 2.6660 - learning_rate: 5.0000e-05
Epoch 56/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2384 - loss: 2.3538 - val_accuracy: 0.1471 -
val_loss: 2.6658 - learning_rate: 5.0000e-05
Epoch 57/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2503 - loss: 2.3736 - val_accuracy: 0.1471 -
val_loss: 2.6608 - learning_rate: 5.0000e-05
Epoch 58/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2581 - loss: 2.3448 - val_accuracy: 0.1507 -
val_loss: 2.6561 - learning_rate: 5.0000e-05
Epoch 59/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2584 - loss: 2.3602 - val_accuracy: 0.1544 -
val_loss: 2.6576 - learning_rate: 5.0000e-05
Epoch 60/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2544 - loss: 2.3640 - val_accuracy: 0.1728 -
val_loss: 2.6513 - learning_rate: 5.0000e-05
Epoch 61/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2565 - loss: 2.3456 - val_accuracy: 0.1728 -
val_loss: 2.6532 - learning_rate: 5.0000e-05
Epoch 62/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2548 - loss: 2.3449 - val_accuracy: 0.1728 -
val_loss: 2.6588 - learning_rate: 5.0000e-05
Epoch 63/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2521 - loss: 2.3356 - val_accuracy: 0.1765 -
val_loss: 2.6521 - learning_rate: 5.0000e-05
Epoch 64/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2624 - loss: 2.3349 - val_accuracy: 0.1765 -
val_loss: 2.6532 - learning_rate: 5.0000e-05
Epoch 65/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2652 - loss: 2.3528 - val_accuracy: 0.1765 -
val_loss: 2.6476 - learning_rate: 5.0000e-05
Epoch 66/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2545 - loss: 2.3774 - val_accuracy: 0.1765 -
val_loss: 2.6509 - learning_rate: 5.0000e-05
Epoch 67/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2656 - loss: 2.3303 - val_accuracy: 0.1765 -
val_loss: 2.6451 - learning_rate: 5.0000e-05
Epoch 68/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2495 - loss: 2.3503 - val_accuracy: 0.1654 -
val_loss: 2.6471 - learning_rate: 5.0000e-05
Epoch 69/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2735 - loss: 2.3421 - val_accuracy: 0.1728 -
val_loss: 2.6399 - learning_rate: 5.0000e-05
Epoch 70/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2555 - loss: 2.3395 - val_accuracy: 0.1618 -
val_loss: 2.6447 - learning_rate: 5.0000e-05
Epoch 71/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2573 - loss: 2.3421 - val_accuracy: 0.1618 -
val_loss: 2.6455 - learning_rate: 5.0000e-05
Epoch 72/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2617 - loss: 2.3510 - val_accuracy: 0.1618 -
val_loss: 2.6417 - learning_rate: 5.0000e-05
Epoch 73/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2509 - loss: 2.3569 - val_accuracy: 0.1728 -
val_loss: 2.6409 - learning_rate: 5.0000e-05
Epoch 74/150
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2522 - loss: 2.3507 - val_accuracy: 0.1618 -
val_loss: 2.6398 - learning_rate: 5.0000e-05
Epoch 75/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 17s/step - accuracy: 0.2647 - loss: 2.3557 - val_accuracy: 0.1581 -
val_loss: 2.6406 - learning_rate: 5.0000e-05
Epoch 76/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2702 - loss: 2.3313 - val_accuracy: 0.1691 -
val_loss: 2.6376 - learning_rate: 5.0000e-05
Epoch 77/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2490 - loss: 2.3429 - val_accuracy: 0.1691 -
val_loss: 2.6298 - learning_rate: 5.0000e-05
Epoch 78/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2614 - loss: 2.3455 - val_accuracy: 0.1654 -
val_loss: 2.6335 - learning_rate: 5.0000e-05
Epoch 79/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2676 - loss: 2.3326 - val_accuracy: 0.1728 -
val_loss: 2.6360 - learning_rate: 5.0000e-05
Epoch 80/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2621 - loss: 2.3445 - val_accuracy: 0.1544 -
val_loss: 2.6330 - learning_rate: 5.0000e-05
Epoch 81/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2432 - loss: 2.3269 - val_accuracy: 0.1728 -
val_loss: 2.6291 - learning_rate: 5.0000e-05
Epoch 82/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2469 - loss: 2.3343 - val_accuracy: 0.1618 -
val_loss: 2.6309 - learning_rate: 5.0000e-05
Epoch 83/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2562 - loss: 2.3467 - val_accuracy: 0.1544 -
val_loss: 2.6299 - learning_rate: 5.0000e-05
Epoch 84/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 12s/step - accuracy: 0.2710 - loss: 2.3331 - val_accuracy: 0.1581 -
val_loss: 2.6305 - learning_rate: 5.0000e-05
Epoch 85/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2797 - loss: 2.2949 - val_accuracy: 0.1728 -
val_loss: 2.6249 - learning_rate: 5.0000e-05
Epoch 86/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2689 - loss: 2.3422 - val_accuracy: 0.1618 -
val_loss: 2.6345 - learning_rate: 5.0000e-05
Epoch 87/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2485 - loss: 2.3309 - val_accuracy: 0.1654 -
val_loss: 2.6281 - learning_rate: 5.0000e-05
Epoch 88/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2575 - loss: 2.3382 - val_accuracy: 0.1507 -
val_loss: 2.6286 - learning_rate: 5.0000e-05
Epoch 89/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2640 - loss: 2.3304 - val_accuracy: 0.1581 -
val_loss: 2.6238 - learning_rate: 5.0000e-05
Epoch 90/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2532 - loss: 2.3397 - val_accuracy: 0.1581 -
val_loss: 2.6193 - learning_rate: 5.0000e-05
Epoch 91/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2540 - loss: 2.3219 - val_accuracy: 0.1654 -
val_loss: 2.6222 - learning_rate: 5.0000e-05
Epoch 92/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 38s 17s/step - accuracy: 0.2583 - loss: 2.3352 - val_accuracy: 0.1581 -
val_loss: 2.6172 - learning_rate: 5.0000e-05
Epoch 93/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2483 - loss: 2.3648 - val_accuracy: 0.1544 -
val_loss: 2.6176 - learning_rate: 5.0000e-05
```

```
Epoch 94/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2612 - loss: 2.3620 - val_accuracy: 0.1618 -
val_loss: 2.6163 - learning_rate: 5.0000e-05
Epoch 95/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2494 - loss: 2.3387 - val_accuracy: 0.1618 -
val_loss: 2.6151 - learning_rate: 5.0000e-05
Epoch 96/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2777 - loss: 2.3399 - val_accuracy: 0.1581 -
val_loss: 2.6158 - learning_rate: 5.0000e-05
Epoch 97/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2473 - loss: 2.3259 - val_accuracy: 0.1581 -
val_loss: 2.6103 - learning_rate: 5.0000e-05
Epoch 98/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2765 - loss: 2.3066 - val_accuracy: 0.1618 -
val_loss: 2.6104 - learning_rate: 5.0000e-05
Epoch 99/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2706 - loss: 2.3127 - val_accuracy: 0.1618 -
val_loss: 2.6054 - learning_rate: 5.0000e-05
Epoch 100/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2827 - loss: 2.2799 - val_accuracy: 0.1581 -
val_loss: 2.6064 - learning_rate: 5.0000e-05
Epoch 101/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2674 - loss: 2.3274 - val_accuracy: 0.1654 -
val_loss: 2.6078 - learning_rate: 5.0000e-05
Epoch 102/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2561 - loss: 2.3280 - val_accuracy: 0.1581 -
val_loss: 2.6000 - learning_rate: 5.0000e-05
Epoch 103/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2654 - loss: 2.2990 - val_accuracy: 0.1544 -
val_loss: 2.6028 - learning_rate: 5.0000e-05
Epoch 104/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2771 - loss: 2.3187 - val_accuracy: 0.1691 -
val_loss: 2.6016 - learning_rate: 5.0000e-05
Epoch 105/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2517 - loss: 2.3233 - val_accuracy: 0.1654 -
val_loss: 2.5945 - learning_rate: 5.0000e-05
Epoch 106/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2730 - loss: 2.3182 - val_accuracy: 0.1728 -
val_loss: 2.5941 - learning_rate: 5.0000e-05
Epoch 107/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2559 - loss: 2.3259 - val_accuracy: 0.1654 -
val_loss: 2.5959 - learning_rate: 5.0000e-05
Epoch 108/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2593 - loss: 2.3188 - val_accuracy: 0.1765 -
val_loss: 2.5856 - learning_rate: 5.0000e-05
Epoch 109/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2788 - loss: 2.3069 - val_accuracy: 0.1728 -
val_loss: 2.5880 - learning_rate: 5.0000e-05
Epoch 110/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2701 - loss: 2.3345 - val_accuracy: 0.1691 -
val_loss: 2.5828 - learning_rate: 5.0000e-05
Epoch 111/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2471 - loss: 2.3320 - val_accuracy: 0.1728 -
val_loss: 2.5816 - learning_rate: 5.0000e-05
Epoch 112/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2838 - loss: 2.2928 - val_accuracy: 0.1581 -
val_loss: 2.5831 - learning_rate: 5.0000e-05
Epoch 113/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2956 - loss: 2.2754 - val_accuracy: 0.1654 -
```

```
val_loss: 2.5807 - learning_rate: 5.0000e-05
Epoch 114/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2593 - loss: 2.3413 - val_accuracy: 0.1728 -
val_loss: 2.5778 - learning_rate: 5.0000e-05
Epoch 115/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2629 - loss: 2.3249 - val_accuracy: 0.1875 -
val_loss: 2.5742 - learning_rate: 5.0000e-05
Epoch 116/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2575 - loss: 2.3040 - val_accuracy: 0.1728 -
val_loss: 2.5818 - learning_rate: 5.0000e-05
Epoch 117/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2728 - loss: 2.3302 - val_accuracy: 0.1728 -
val_loss: 2.5747 - learning_rate: 5.0000e-05
Epoch 118/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2657 - loss: 2.3192 - val_accuracy: 0.1544 -
val_loss: 2.5744 - learning_rate: 5.0000e-05
Epoch 119/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2985 - loss: 2.3011 - val_accuracy: 0.1728 -
val_loss: 2.5697 - learning_rate: 5.0000e-05
Epoch 120/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2592 - loss: 2.3219 - val_accuracy: 0.1618 -
val_loss: 2.5651 - learning_rate: 5.0000e-05
Epoch 121/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2831 - loss: 2.2954 - val_accuracy: 0.1654 -
val_loss: 2.5604 - learning_rate: 5.0000e-05
Epoch 122/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2742 - loss: 2.3089 - val_accuracy: 0.1728 -
val_loss: 2.5630 - learning_rate: 5.0000e-05
Epoch 123/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2684 - loss: 2.3235 - val_accuracy: 0.1654 -
val_loss: 2.5553 - learning_rate: 5.0000e-05
Epoch 124/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2563 - loss: 2.3193 - val_accuracy: 0.1654 -
val_loss: 2.5539 - learning_rate: 5.0000e-05
Epoch 125/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2718 - loss: 2.3187 - val_accuracy: 0.1728 -
val_loss: 2.5496 - learning_rate: 5.0000e-05
Epoch 126/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2681 - loss: 2.3160 - val_accuracy: 0.1654 -
val_loss: 2.5533 - learning_rate: 5.0000e-05
Epoch 127/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2585 - loss: 2.3240 - val_accuracy: 0.1691 -
val_loss: 2.5524 - learning_rate: 5.0000e-05
Epoch 128/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2748 - loss: 2.3028 - val_accuracy: 0.1691 -
val_loss: 2.5514 - learning_rate: 5.0000e-05
Epoch 129/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2847 - loss: 2.2594 - val_accuracy: 0.1728 -
val_loss: 2.5458 - learning_rate: 5.0000e-05
Epoch 130/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2811 - loss: 2.3160 - val_accuracy: 0.1728 -
val_loss: 2.5474 - learning_rate: 5.0000e-05
Epoch 131/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2566 - loss: 2.3209 - val_accuracy: 0.1765 -
val_loss: 2.5474 - learning_rate: 5.0000e-05
Epoch 132/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2765 - loss: 2.3149 - val_accuracy: 0.1691 -
val_loss: 2.5439 - learning_rate: 5.0000e-05
Epoch 133/150
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2813 - loss: 2.2979 - val_accuracy: 0.1691 -
val_loss: 2.5429 - learning_rate: 5.0000e-05
Epoch 134/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2724 - loss: 2.3216 - val_accuracy: 0.1801 -
val_loss: 2.5308 - learning_rate: 5.0000e-05
Epoch 135/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2794 - loss: 2.2960 - val_accuracy: 0.1838 -
val_loss: 2.5300 - learning_rate: 5.0000e-05
Epoch 136/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2548 - loss: 2.2959 - val_accuracy: 0.1765 -
val_loss: 2.5270 - learning_rate: 5.0000e-05
Epoch 137/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2643 - loss: 2.3167 - val_accuracy: 0.1765 -
val_loss: 2.5195 - learning_rate: 5.0000e-05
Epoch 138/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2780 - loss: 2.2877 - val_accuracy: 0.1912 -
val_loss: 2.5187 - learning_rate: 5.0000e-05
Epoch 139/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2587 - loss: 2.3560 - val_accuracy: 0.1728 -
val_loss: 2.5215 - learning_rate: 5.0000e-05
Epoch 140/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2824 - loss: 2.2929 - val_accuracy: 0.1985 -
val_loss: 2.5115 - learning_rate: 5.0000e-05
Epoch 141/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2547 - loss: 2.3148 - val_accuracy: 0.1875 -
val_loss: 2.5110 - learning_rate: 5.0000e-05
Epoch 142/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2923 - loss: 2.2751 - val_accuracy: 0.1912 -
val_loss: 2.5120 - learning_rate: 5.0000e-05
Epoch 143/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2785 - loss: 2.3117 - val_accuracy: 0.1912 -
val_loss: 2.5140 - learning_rate: 5.0000e-05
Epoch 144/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2532 - loss: 2.3190 - val_accuracy: 0.1875 -
val_loss: 2.5035 - learning_rate: 5.0000e-05
Epoch 145/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2846 - loss: 2.3042 - val_accuracy: 0.1949 -
val_loss: 2.4982 - learning_rate: 5.0000e-05
Epoch 146/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 17s/step - accuracy: 0.2625 - loss: 2.3244 - val_accuracy: 0.2022 -
val_loss: 2.5058 - learning_rate: 5.0000e-05
Epoch 147/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2693 - loss: 2.3237 - val_accuracy: 0.1985 -
val_loss: 2.4971 - learning_rate: 5.0000e-05
Epoch 148/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2837 - loss: 2.2912 - val_accuracy: 0.1912 -
val_loss: 2.4992 - learning_rate: 5.0000e-05
Epoch 149/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2706 - loss: 2.2968 - val_accuracy: 0.2169 -
val_loss: 2.4896 - learning_rate: 5.0000e-05
Epoch 150/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2689 - loss: 2.3184 - val_accuracy: 0.2132 -
val_loss: 2.4877 - learning_rate: 5.0000e-05
```

In [52]:

```python
#check metrics on test data and plot accuracy curve

animal_acc_3 = history_CNN_3.history['accuracy']
animal_val_acc_3 = history_CNN_3.history['val_accuracy']

epochs_ind = [i for i in range(1, 1 + epochs)]
```

```
plt.plot(epochs_ind, animal_acc_3, 'blue', label = 'Training accuracy')
plt.plot(epochs_ind, animal_val_acc_3, 'red', label = 'Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy over training epochs')
plt.legend()
plt.show()
```

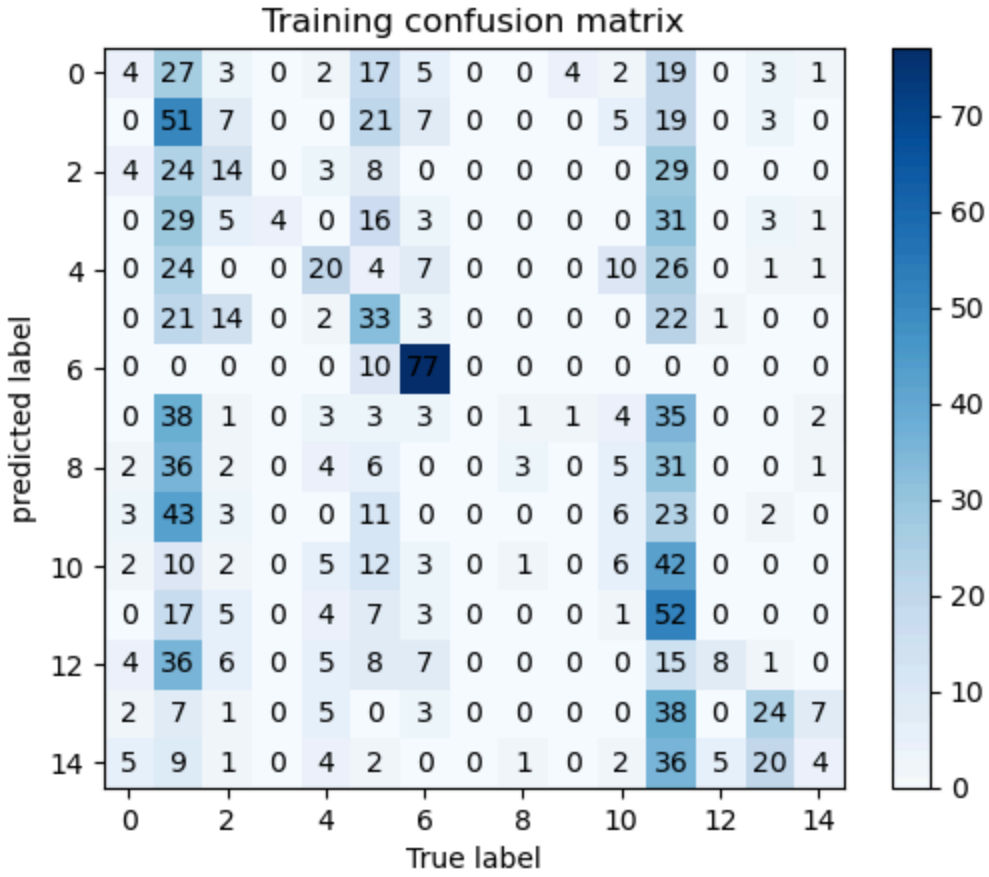
In [53]:
```
_, train_acc = animal_CNN_3.evaluate(X_train, Y_train, verbose = 0)
_, test_acc = animal_CNN_3.evaluate(X_test, Y_test, verbose = 0)

print(f"Training accuracy: {train_acc*100}%")
print(f"Testing accuracy: {test_acc*100}%")
```

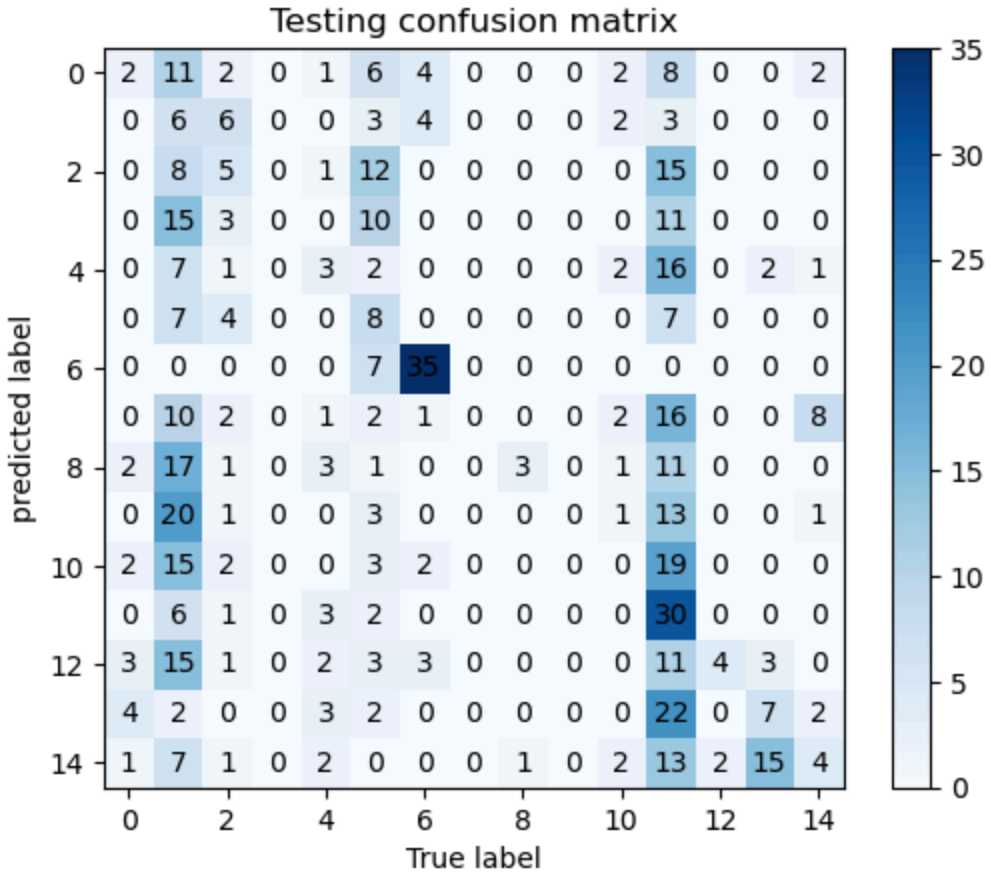
```
Training accuracy: 22.058823704719543%
Testing accuracy: 18.321917951107025%
```

In [54]:
```
#confusion matrix

train_preds = np.argmax(animal_CNN_3.predict(X_train), axis = 1)
test_preds = np.argmax(animal_CNN_3.predict(X_test), axis = 1)

train_cm = confusion_matrix(np.argmax(Y_train, axis = 1), train_preds)
test_cm = confusion_matrix(np.argmax(Y_test, axis = 1), test_preds)

plt.imshow(train_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(train_cm.shape[0]):
  for j in range(train_cm.shape[1]):
    plt.text(j, i, train_cm[i, j], ha = 'center', va = 'center')
plt.title('Training confusion matrix')
```

```
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()

plt.imshow(test_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(test_cm.shape[0]):
  for j in range(test_cm.shape[1]):
    plt.text(j, i, test_cm[i, j], ha = 'center', va = 'center')
plt.title('Testing confusion matrix')
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 4s 92ms/step
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 86ms/step
```





This appears to be performing better, but is still underperforming. The model starts with a large separation between the train and validate sets, which gradually converge towards each other as seen in the learning curves, I note the behavior that there are points where the training curve will decrease while the validation curve increases, indicating that the network is attempting to compensate for both at the same time, allowing for it to generalize during training. I will add more layers to increase accuracy, while adding slightly more dropout and more regularization to compensate for the potential for overfitting with more layers. Finally I will try slightly decreasing the batch size, and increasing the number of epochs to 200, while have the learning rate decay more slowly. If this does not work I will attempt to implement transfer learning onto this dataset, using the MobileNetV2 model from Keras applications to improve the accuracy over what can be normally achieved without significantly more computing power.

In [55]:
```
#new model architecture
animal_CNN_4 = Sequential()
animal_CNN_4.add(InputLayer(input_shape = X_shape[1:]))
animal_CNN_4.add(Conv2D(filters = 64,
                        kernel_size = (3, 3),
                        activation = None,
                        padding = 'same',
                        kernel_regularizer = regularizers.l1_l2(0.0005, 0.0005)))
animal_CNN_4.add(BatchNormalization())
animal_CNN_4.add(Activation('relu'))
animal_CNN_4.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_4.add(Dropout(0.25))

animal_CNN_4.add(Conv2D(filters = 64,
                        kernel_size = (3, 3),
                        activation = None,
                        padding = 'same',
```

```
                         kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_4.add(BatchNormalization())
animal_CNN_4.add(Activation('relu'))
animal_CNN_4.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_4.add(Dropout(0.25))

animal_CNN_4.add(Conv2D(filters = 32,
                        kernel_size = (3, 3),
                        padding = 'same',
                        activation = None,
                        kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_4.add(BatchNormalization())
animal_CNN_4.add(Activation('relu'))
animal_CNN_4.add(MaxPooling2D(pool_size = (2,2)))
animal_CNN_4.add(Dropout(0.25))

animal_CNN_4.add(GlobalAveragePooling2D())
animal_CNN_4.add(Dense(64,
                       activation = 'relu',
                       kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_4.add(Dropout(0.1))

animal_CNN_4.add(Dense(16,
                       activation = 'relu',
                       kernel_regularizer = regularizers.l2(0.0005)))
animal_CNN_4.add(Dropout(0.1))

animal_CNN_4.add(Dense(Y_shape[1], activation = 'softmax'))

optimizer = Adam(learning_rate = 0.001, amsgrad = True)

animal_CNN_4.compile(loss = 'categorical_crossentropy',
                     optimizer = optimizer,
                     metrics = ['accuracy'])

animal_CNN_4.summary()
```

```
C:\Users\lcleymaet\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\core\input_lay
er.py:27: UserWarning: Argument `input_shape` is deprecated. Use `shape` instead.
  warnings.warn(
```

Model: "sequential_11"

Layer (type)	Output Shape	Param #
conv2d_29 (Conv2D)	(None, 224, 224, 64)	1,792
batch_normalization_29 (BatchNormalization)	(None, 224, 224, 64)	256
activation_29 (Activation)	(None, 224, 224, 64)	0
max_pooling2d_29 (MaxPooling2D)	(None, 112, 112, 64)	0
dropout_41 (Dropout)	(None, 112, 112, 64)	0
conv2d_30 (Conv2D)	(None, 112, 112, 64)	36,928
batch_normalization_30 (BatchNormalization)	(None, 112, 112, 64)	256
activation_30 (Activation)	(None, 112, 112, 64)	0
max_pooling2d_30 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_42 (Dropout)	(None, 56, 56, 64)	0
conv2d_31 (Conv2D)	(None, 56, 56, 32)	18,464
batch_normalization_31 (BatchNormalization)	(None, 56, 56, 32)	128
activation_31 (Activation)	(None, 56, 56, 32)	0
max_pooling2d_31 (MaxPooling2D)	(None, 28, 28, 32)	0
dropout_43 (Dropout)	(None, 28, 28, 32)	0
global_average_pooling2d_8 (GlobalAveragePooling2D)	(None, 32)	0
dense_23 (Dense)	(None, 64)	2,112
dropout_44 (Dropout)	(None, 64)	0
dense_24 (Dense)	(None, 16)	1,040
dropout_45 (Dropout)	(None, 16)	0
dense_25 (Dense)	(None, 15)	255

Total params: 61,231 (239.18 KB)

Trainable params: 60,911 (237.93 KB)

Non-trainable params: 320 (1.25 KB)

In [56]:
```
#reduces learning rate if loss value plateaus for 3 iterations
reduce_2 = ReduceLROnPlateau(
    monitor = 'val_loss',
    patience = 3,
    verbose = 1,
```

```
    factor = 0.6,
    min_lr = 0.000005
)

#train the model
epochs = 200
batch_size = 32

history_CNN_4 = animal_CNN_4.fit(train_gen,
                                 epochs = epochs,
                                 batch_size = batch_size,
                                validation_data = val_gen,
                               callbacks = [reduce_2])
```

```
Epoch 1/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 48s 19s/step - accuracy: 0.0672 - loss: 2.9265 - val_accuracy: 0.0478 -
val_loss: 2.8405 - learning_rate: 0.0010
Epoch 2/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.0914 - loss: 2.8346 - val_accuracy: 0.0588 -
val_loss: 2.8394 - learning_rate: 0.0010
Epoch 3/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.0985 - loss: 2.7757 - val_accuracy: 0.0772 -
val_loss: 2.8383 - learning_rate: 0.0010
Epoch 4/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1072 - loss: 2.7786 - val_accuracy: 0.1103 -
val_loss: 2.8362 - learning_rate: 0.0010
Epoch 5/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 17s/step - accuracy: 0.1423 - loss: 2.7613 - val_accuracy: 0.1140 -
val_loss: 2.8345 - learning_rate: 0.0010
Epoch 6/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1317 - loss: 2.7587 - val_accuracy: 0.0993 -
val_loss: 2.8323 - learning_rate: 0.0010
Epoch 7/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.1204 - loss: 2.7459 - val_accuracy: 0.1066 -
val_loss: 2.8303 - learning_rate: 0.0010
Epoch 8/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1535 - loss: 2.7227 - val_accuracy: 0.1103 -
val_loss: 2.8279 - learning_rate: 0.0010
Epoch 9/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1456 - loss: 2.7241 - val_accuracy: 0.1029 -
val_loss: 2.8252 - learning_rate: 0.0010
Epoch 10/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1523 - loss: 2.7203 - val_accuracy: 0.1029 -
val_loss: 2.8235 - learning_rate: 0.0010
Epoch 11/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1650 - loss: 2.6975 - val_accuracy: 0.1103 -
val_loss: 2.8219 - learning_rate: 0.0010
Epoch 12/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1656 - loss: 2.6855 - val_accuracy: 0.1066 -
val_loss: 2.8201 - learning_rate: 0.0010
Epoch 13/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1770 - loss: 2.6747 - val_accuracy: 0.1066 -
val_loss: 2.8189 - learning_rate: 0.0010
Epoch 14/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1678 - loss: 2.6645 - val_accuracy: 0.1029 -
val_loss: 2.8175 - learning_rate: 0.0010
Epoch 15/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1601 - loss: 2.6446 - val_accuracy: 0.0993 -
val_loss: 2.8165 - learning_rate: 0.0010
Epoch 16/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.1746 - loss: 2.6446 - val_accuracy: 0.1029 -
val_loss: 2.8166 - learning_rate: 0.0010
Epoch 17/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1823 - loss: 2.6329 - val_accuracy: 0.0956 -
val_loss: 2.8128 - learning_rate: 0.0010
Epoch 18/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2076 - loss: 2.5941 - val_accuracy: 0.1029 -
val_loss: 2.8136 - learning_rate: 0.0010
Epoch 19/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2037 - loss: 2.5740 - val_accuracy: 0.0956 -
val_loss: 2.8158 - learning_rate: 0.0010
Epoch 20/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.1797 - loss: 2.6206
```

```
Epoch 20: ReduceLROnPlateau reducing learning rate to 0.0006000000284984708.
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1814 - loss: 2.6143 - val_accuracy: 0.0956 -
val_loss: 2.8148 - learning_rate: 0.0010
Epoch 21/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2106 - loss: 2.5584 - val_accuracy: 0.1029 -
val_loss: 2.8165 - learning_rate: 6.0000e-04
Epoch 22/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2047 - loss: 2.5898 - val_accuracy: 0.0662 -
val_loss: 2.8185 - learning_rate: 6.0000e-04
Epoch 23/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 15s/step - accuracy: 0.2237 - loss: 2.5483
Epoch 23: ReduceLROnPlateau reducing learning rate to 0.0003600000170990825.
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2231 - loss: 2.5514 - val_accuracy: 0.0625 -
val_loss: 2.8203 - learning_rate: 6.0000e-04
Epoch 24/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.1997 - loss: 2.5629 - val_accuracy: 0.0662 -
val_loss: 2.8210 - learning_rate: 3.6000e-04
Epoch 25/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1865 - loss: 2.5808 - val_accuracy: 0.0662 -
val_loss: 2.8209 - learning_rate: 3.6000e-04
Epoch 26/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.2193 - loss: 2.5296
Epoch 26: ReduceLROnPlateau reducing learning rate to 0.00021600000327453016.
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2189 - loss: 2.5318 - val_accuracy: 0.0625 -
val_loss: 2.8218 - learning_rate: 3.6000e-04
Epoch 27/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2350 - loss: 2.5303 - val_accuracy: 0.0735 -
val_loss: 2.8218 - learning_rate: 2.1600e-04
Epoch 28/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2321 - loss: 2.5319 - val_accuracy: 0.0735 -
val_loss: 2.8223 - learning_rate: 2.1600e-04
Epoch 29/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 14s/step - accuracy: 0.2134 - loss: 2.5448
Epoch 29: ReduceLROnPlateau reducing learning rate to 0.00012960000021848827.
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2150 - loss: 2.5426 - val_accuracy: 0.0735 -
val_loss: 2.8228 - learning_rate: 2.1600e-04
Epoch 30/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2235 - loss: 2.5391 - val_accuracy: 0.0735 -
val_loss: 2.8230 - learning_rate: 1.2960e-04
Epoch 31/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2278 - loss: 2.5271 - val_accuracy: 0.0699 -
val_loss: 2.8238 - learning_rate: 1.2960e-04
Epoch 32/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.2261 - loss: 2.5302
Epoch 32: ReduceLROnPlateau reducing learning rate to 7.775999838486313e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2266 - loss: 2.5282 - val_accuracy: 0.0662 -
val_loss: 2.8242 - learning_rate: 1.2960e-04
Epoch 33/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2144 - loss: 2.5103 - val_accuracy: 0.0735 -
val_loss: 2.8244 - learning_rate: 7.7760e-05
Epoch 34/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 17s/step - accuracy: 0.2374 - loss: 2.5275 - val_accuracy: 0.0735 -
val_loss: 2.8253 - learning_rate: 7.7760e-05
Epoch 35/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.2227 - loss: 2.5274
Epoch 35: ReduceLROnPlateau reducing learning rate to 5e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2210 - loss: 2.5282 - val_accuracy: 0.0735 -
val_loss: 2.8257 - learning_rate: 7.7760e-05
Epoch 36/200
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2179 - loss: 2.5322 - val_accuracy: 0.0699 -
val_loss: 2.8268 - learning_rate: 5.0000e-05
Epoch 37/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2251 - loss: 2.5064 - val_accuracy: 0.0662 -
val_loss: 2.8276 - learning_rate: 5.0000e-05
Epoch 38/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2259 - loss: 2.5143 - val_accuracy: 0.0772 -
val_loss: 2.8277 - learning_rate: 5.0000e-05
Epoch 39/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2266 - loss: 2.5052 - val_accuracy: 0.0735 -
val_loss: 2.8282 - learning_rate: 5.0000e-05
Epoch 40/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2212 - loss: 2.5226 - val_accuracy: 0.0772 -
val_loss: 2.8290 - learning_rate: 5.0000e-05
Epoch 41/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2180 - loss: 2.5180 - val_accuracy: 0.0772 -
val_loss: 2.8294 - learning_rate: 5.0000e-05
Epoch 42/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2200 - loss: 2.5526 - val_accuracy: 0.0772 -
val_loss: 2.8301 - learning_rate: 5.0000e-05
Epoch 43/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2328 - loss: 2.5111 - val_accuracy: 0.0772 -
val_loss: 2.8307 - learning_rate: 5.0000e-05
Epoch 44/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2251 - loss: 2.5239 - val_accuracy: 0.0735 -
val_loss: 2.8313 - learning_rate: 5.0000e-05
Epoch 45/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2362 - loss: 2.4990 - val_accuracy: 0.0662 -
val_loss: 2.8325 - learning_rate: 5.0000e-05
Epoch 46/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2328 - loss: 2.5091 - val_accuracy: 0.0735 -
val_loss: 2.8327 - learning_rate: 5.0000e-05
Epoch 47/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2243 - loss: 2.5113 - val_accuracy: 0.0735 -
val_loss: 2.8330 - learning_rate: 5.0000e-05
Epoch 48/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2271 - loss: 2.5172 - val_accuracy: 0.0809 -
val_loss: 2.8333 - learning_rate: 5.0000e-05
Epoch 49/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2457 - loss: 2.4957 - val_accuracy: 0.0699 -
val_loss: 2.8338 - learning_rate: 5.0000e-05
Epoch 50/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2309 - loss: 2.5146 - val_accuracy: 0.0699 -
val_loss: 2.8344 - learning_rate: 5.0000e-05
Epoch 51/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2294 - loss: 2.5166 - val_accuracy: 0.0662 -
val_loss: 2.8348 - learning_rate: 5.0000e-05
Epoch 52/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2193 - loss: 2.5196 - val_accuracy: 0.0699 -
val_loss: 2.8348 - learning_rate: 5.0000e-05
Epoch 53/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2344 - loss: 2.4908 - val_accuracy: 0.0662 -
val_loss: 2.8360 - learning_rate: 5.0000e-05
Epoch 54/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2212 - loss: 2.5128 - val_accuracy: 0.0699 -
val_loss: 2.8358 - learning_rate: 5.0000e-05
Epoch 55/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2307 - loss: 2.5068 - val_accuracy: 0.0772 -
val_loss: 2.8356 - learning_rate: 5.0000e-05
```

```
Epoch 56/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2350 - loss: 2.4842 - val_accuracy: 0.0809 -
val_loss: 2.8350 - learning_rate: 5.0000e-05
Epoch 57/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2071 - loss: 2.5129 - val_accuracy: 0.0699 -
val_loss: 2.8362 - learning_rate: 5.0000e-05
Epoch 58/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2580 - loss: 2.4744 - val_accuracy: 0.0662 -
val_loss: 2.8357 - learning_rate: 5.0000e-05
Epoch 59/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1912 - loss: 2.5393 - val_accuracy: 0.0662 -
val_loss: 2.8356 - learning_rate: 5.0000e-05
Epoch 60/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2270 - loss: 2.5152 - val_accuracy: 0.0735 -
val_loss: 2.8361 - learning_rate: 5.0000e-05
Epoch 61/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2228 - loss: 2.5016 - val_accuracy: 0.0699 -
val_loss: 2.8358 - learning_rate: 5.0000e-05
Epoch 62/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 17s/step - accuracy: 0.2304 - loss: 2.4822 - val_accuracy: 0.0735 -
val_loss: 2.8354 - learning_rate: 5.0000e-05
Epoch 63/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 40s 11s/step - accuracy: 0.2178 - loss: 2.4934 - val_accuracy: 0.0735 -
val_loss: 2.8362 - learning_rate: 5.0000e-05
Epoch 64/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.1979 - loss: 2.5032 - val_accuracy: 0.0735 -
val_loss: 2.8350 - learning_rate: 5.0000e-05
Epoch 65/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2273 - loss: 2.4958 - val_accuracy: 0.0772 -
val_loss: 2.8355 - learning_rate: 5.0000e-05
Epoch 66/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2021 - loss: 2.5397 - val_accuracy: 0.0772 -
val_loss: 2.8352 - learning_rate: 5.0000e-05
Epoch 67/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2202 - loss: 2.4970 - val_accuracy: 0.0772 -
val_loss: 2.8344 - learning_rate: 5.0000e-05
Epoch 68/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2320 - loss: 2.4854 - val_accuracy: 0.0772 -
val_loss: 2.8348 - learning_rate: 5.0000e-05
Epoch 69/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2356 - loss: 2.4936 - val_accuracy: 0.0735 -
val_loss: 2.8348 - learning_rate: 5.0000e-05
Epoch 70/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2317 - loss: 2.4834 - val_accuracy: 0.0735 -
val_loss: 2.8346 - learning_rate: 5.0000e-05
Epoch 71/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2311 - loss: 2.4926 - val_accuracy: 0.0809 -
val_loss: 2.8346 - learning_rate: 5.0000e-05
Epoch 72/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 11s/step - accuracy: 0.2478 - loss: 2.4772 - val_accuracy: 0.0735 -
val_loss: 2.8350 - learning_rate: 5.0000e-05
Epoch 73/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2071 - loss: 2.5291 - val_accuracy: 0.0772 -
val_loss: 2.8348 - learning_rate: 5.0000e-05
Epoch 74/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2451 - loss: 2.4788 - val_accuracy: 0.0772 -
val_loss: 2.8352 - learning_rate: 5.0000e-05
Epoch 75/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2172 - loss: 2.4956 - val_accuracy: 0.0809 -
```

```
val_loss: 2.8348 - learning_rate: 5.0000e-05
Epoch 76/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2234 - loss: 2.5036 - val_accuracy: 0.0846 -
val_loss: 2.8338 - learning_rate: 5.0000e-05
Epoch 77/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2307 - loss: 2.4751 - val_accuracy: 0.0846 -
val_loss: 2.8345 - learning_rate: 5.0000e-05
Epoch 78/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2222 - loss: 2.5272 - val_accuracy: 0.0846 -
val_loss: 2.8338 - learning_rate: 5.0000e-05
Epoch 79/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2349 - loss: 2.4715 - val_accuracy: 0.0772 -
val_loss: 2.8340 - learning_rate: 5.0000e-05
Epoch 80/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2513 - loss: 2.4796 - val_accuracy: 0.0772 -
val_loss: 2.8345 - learning_rate: 5.0000e-05
Epoch 81/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2276 - loss: 2.4839 - val_accuracy: 0.0772 -
val_loss: 2.8342 - learning_rate: 5.0000e-05
Epoch 82/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 36s 10s/step - accuracy: 0.2313 - loss: 2.4851 - val_accuracy: 0.0772 -
val_loss: 2.8328 - learning_rate: 5.0000e-05
Epoch 83/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 10s/step - accuracy: 0.2445 - loss: 2.4845 - val_accuracy: 0.0735 -
val_loss: 2.8326 - learning_rate: 5.0000e-05
Epoch 84/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 37s 17s/step - accuracy: 0.2259 - loss: 2.4836 - val_accuracy: 0.0809 -
val_loss: 2.8318 - learning_rate: 5.0000e-05
Epoch 85/200
1/3 ━━━━━━━━━━━━━━━━━━━━ 6s 3s/step - accuracy: 0.2500 - loss: 2.3935
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[56], line 14
     11 epochs = 200
     12 batch_size = 32
---> 14 history_CNN_4 = animal_CNN_4.fit(train_gen,
     15                                  epochs = epochs,
     16                                  batch_size = batch_size,
     17                                 validation_data = val_gen,
     18                                callbacks = [reduce_2])

File ~\AppData\Roaming\Python\Python312\site-packages\keras\src\utils\traceback_utils.py:117, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\AppData\Roaming\Python\Python312\site-packages\keras\src\backend\tensorflow\trainer.py:371, in TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_freq)
    369 for step, iterator in epoch_iterator:
    370     callbacks.on_train_batch_begin(step)
--> 371     logs = self.train_function(iterator)
    372     callbacks.on_train_batch_end(step, logs)
    373     if self.stop_training:

File ~\AppData\Roaming\Python\Python312\site-packages\keras\src\backend\tensorflow\trainer.py:219, in TensorFlowTrainer._make_function.<locals>.function(iterator)
    215 def function(iterator):
    216     if isinstance(
    217         iterator, (tf.data.Iterator, tf.distribute.DistributedIterator)
    218     ):
--> 219         opt_outputs = multi_step_on_iterator(iterator)
    220         if not opt_outputs.has_value():
    221             raise StopIteration

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\util\traceback_utils.py:150, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_function\polymorphic_function.py:833, in Function.__call__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_function\polymorphic_function.py:878, in Function._call(self, *args, **kwds)
    875 self._lock.release()
    876 # In this case we have not created variables on the first call. So we can
```

```
    877 # run the first trace but we should fail if variables are created.
--> 878 results = tracing_compilation.call_function(
    879     args, kwds, self._variable_creation_config
    880 )
    881 if self._created_variables:
    882   raise ValueError("Creating variables on a non-first call to a function"
    883                    " decorated with tf.function.")

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_function\tracing_compilation.py:139, in call_function(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_function\concrete_function.py:1322, in ConcreteFunction._call_flat(self, tensor_inputs, captured_inputs)
   1318 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1319 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1320     and executing_eagerly):
   1321   # No tape is watching; skip to running the function.
-> 1322   return self._inference_function.call_preflattened(args)
   1323 forward_backward = self._select_forward_and_backward_functions(
   1324     args,
   1325     possible_gradient_type,
   1326     executing_eagerly)
   1327 forward_function, args_with_tangents = forward_backward.forward()

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_function\atomic_function.py:216, in AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) -> Any:
    215   """Calls with flattened tensor inputs and returns the structured output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_function\atomic_function.py:251, in AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\context.py:1688, in Context.call_function(self, name, tensor_inputs, num_outputs)
   1686 cancellation_context = cancellation.context()
   1687 if cancellation_context is None:
-> 1688   outputs = execute.execute(
   1689       name.decode("utf-8"),
   1690       num_outputs=num_outputs,
```

```
   1691       inputs=tensor_inputs,
   1692       attrs=attrs,
   1693       ctx=self,
   1694   )
   1695 else:
   1696   outputs = execute.execute_with_cancellation(
   1697       name.decode("utf-8"),
   1698       num_outputs=num_outputs,
   (...)
   1702       cancellation_manager=cancellation_context,
   1703   )

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\execute.py:53, in q
uick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

This was interrupted as I noticed the validation data was not breaking away from 1/15 accuracy, thus is simply overfitting, and learning incredibly slowly. To remedy this I will attempt to implement a transfer learning approach.

In [16]:
```
from keras.applications import MobileNetV2
```

In [17]:
```
temp_tx = MobileNetV2(weights = 'imagenet', include_top = True)
```

In [23]:
```
for layer in temp_tx.layers:
    layer.trainable = False

temp_tx.layers[-1].trainable = True

animal_tx_1 = Sequential()
animal_tx_1.add(temp_tx)

animal_tx_1.add(Dense(Y_shape[1], activation = 'softmax'))

optimizer = Adam(learning_rate = 0.001, amsgrad = True)

animal_tx_1.compile(loss = 'categorical_crossentropy',
                    optimizer = optimizer,
                    metrics = ['accuracy'])

animal_tx_1.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 1000)	3,538,984
dense_1 (Dense)	(None, 15)	15,015

Total params: 3,553,999 (13.56 MB)

Trainable params: 1,296,015 (4.94 MB)

Non-trainable params: 2,257,984 (8.61 MB)

In [24]:
```
#reduces learning rate if loss value plateaus for 3 iterations
reduce_2 = ReduceLROnPlateau(
    monitor = 'val_loss',
    patience = 3,
    verbose = 1,
    factor = 0.6,
    min_lr = 0.000005
)

#train the model
epochs = 150
batch_size = 32

history_tx_1 = animal_tx_1.fit(train_gen,
                               epochs = epochs,
                               batch_size = batch_size,
                              validation_data = val_gen,
                             callbacks = [reduce_2])
```

```
C:\Users\lcleymaet\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 43s 9s/step - accuracy: 0.0659 - loss: 2.7096 - val_accuracy: 0.1397 - v
al_loss: 2.7007 - learning_rate: 0.0010
Epoch 2/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.1980 - loss: 2.6979 - val_accuracy: 0.2353 - v
al_loss: 2.6938 - learning_rate: 0.0010
Epoch 3/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.2976 - loss: 2.6906 - val_accuracy: 0.3088 - v
al_loss: 2.6911 - learning_rate: 0.0010
Epoch 4/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.3622 - loss: 2.6843 - val_accuracy: 0.3125 - v
al_loss: 2.6843 - learning_rate: 0.0010
Epoch 5/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.3945 - loss: 2.6784 - val_accuracy: 0.3199 - v
al_loss: 2.6812 - learning_rate: 0.0010
Epoch 6/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.4070 - loss: 2.6720 - val_accuracy: 0.3750 -
val_loss: 2.6766 - learning_rate: 0.0010
Epoch 7/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.4470 - loss: 2.6672 - val_accuracy: 0.4375 - v
al_loss: 2.6690 - learning_rate: 0.0010
Epoch 8/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.4918 - loss: 2.6614 - val_accuracy: 0.4485 - v
al_loss: 2.6632 - learning_rate: 0.0010
Epoch 9/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.5012 - loss: 2.6559 - val_accuracy: 0.4228 - v
al_loss: 2.6600 - learning_rate: 0.0010
Epoch 10/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.5307 - loss: 2.6502 - val_accuracy: 0.4559 - v
al_loss: 2.6557 - learning_rate: 0.0010
Epoch 11/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.5252 - loss: 2.6437 - val_accuracy: 0.4816 -
val_loss: 2.6496 - learning_rate: 0.0010
Epoch 12/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.5648 - loss: 2.6382 - val_accuracy: 0.5625 - v
al_loss: 2.6410 - learning_rate: 0.0010
Epoch 13/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.6361 - loss: 2.6278 - val_accuracy: 0.5368 -
val_loss: 2.6374 - learning_rate: 0.0010
Epoch 14/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.6009 - loss: 2.6267 - val_accuracy: 0.5625 -
val_loss: 2.6312 - learning_rate: 0.0010
Epoch 15/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.6582 - loss: 2.6180 - val_accuracy: 0.5846 - v
al_loss: 2.6237 - learning_rate: 0.0010
Epoch 16/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.6758 - loss: 2.6100 - val_accuracy: 0.5993 - v
al_loss: 2.6169 - learning_rate: 0.0010
Epoch 17/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.6476 - loss: 2.6079 - val_accuracy: 0.5735 - v
al_loss: 2.6163 - learning_rate: 0.0010
Epoch 18/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.6857 - loss: 2.5983 - val_accuracy: 0.6140 - v
al_loss: 2.6056 - learning_rate: 0.0010
Epoch 19/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.6980 - loss: 2.5932 - val_accuracy: 0.6140 - v
al_loss: 2.5980 - learning_rate: 0.0010
Epoch 20/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.6993 - loss: 2.5851 - val_accuracy: 0.6029 - v
```

```
al_loss: 2.5956 - learning_rate: 0.0010
Epoch 21/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.6911 - loss: 2.5805 - val_accuracy: 0.6324 - v
al_loss: 2.5876 - learning_rate: 0.0010
Epoch 22/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.7201 - loss: 2.5707 - val_accuracy: 0.6471 - v
al_loss: 2.5828 - learning_rate: 0.0010
Epoch 23/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.7008 - loss: 2.5659 - val_accuracy: 0.6066 - v
al_loss: 2.5767 - learning_rate: 0.0010
Epoch 24/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.7092 - loss: 2.5598 - val_accuracy: 0.6324 -
val_loss: 2.5696 - learning_rate: 0.0010
Epoch 25/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.7294 - loss: 2.5521 - val_accuracy: 0.6360 - v
al_loss: 2.5663 - learning_rate: 0.0010
Epoch 26/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.7303 - loss: 2.5452 - val_accuracy: 0.6287 - v
al_loss: 2.5600 - learning_rate: 0.0010
Epoch 27/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.7444 - loss: 2.5389 - val_accuracy: 0.6544 - v
al_loss: 2.5501 - learning_rate: 0.0010
Epoch 28/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.7528 - loss: 2.5306 - val_accuracy: 0.6544 -
val_loss: 2.5427 - learning_rate: 0.0010
Epoch 29/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.7543 - loss: 2.5243 - val_accuracy: 0.6765 - v
al_loss: 2.5368 - learning_rate: 0.0010
Epoch 30/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.7291 - loss: 2.5197 - val_accuracy: 0.6397 -
val_loss: 2.5315 - learning_rate: 0.0010
Epoch 31/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.7649 - loss: 2.5073 - val_accuracy: 0.6397 -
val_loss: 2.5269 - learning_rate: 0.0010
Epoch 32/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.7354 - loss: 2.5075 - val_accuracy: 0.6985 - v
al_loss: 2.5192 - learning_rate: 0.0010
Epoch 33/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.7774 - loss: 2.4999 - val_accuracy: 0.7206 - v
al_loss: 2.5114 - learning_rate: 0.0010
Epoch 34/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.7763 - loss: 2.4899 - val_accuracy: 0.7096 -
val_loss: 2.5002 - learning_rate: 0.0010
Epoch 35/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.7991 - loss: 2.4839 - val_accuracy: 0.6912 - v
al_loss: 2.5059 - learning_rate: 0.0010
Epoch 36/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.7847 - loss: 2.4786 - val_accuracy: 0.7243 - v
al_loss: 2.4910 - learning_rate: 0.0010
Epoch 37/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.7782 - loss: 2.4727 - val_accuracy: 0.7022 - v
al_loss: 2.4830 - learning_rate: 0.0010
Epoch 38/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.7813 - loss: 2.4659 - val_accuracy: 0.6985 - v
al_loss: 2.4845 - learning_rate: 0.0010
Epoch 39/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8001 - loss: 2.4588 - val_accuracy: 0.7059 - v
al_loss: 2.4774 - learning_rate: 0.0010
Epoch 40/150
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8067 - loss: 2.4525 - val_accuracy: 0.7390 - v
al_loss: 2.4661 - learning_rate: 0.0010
Epoch 41/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8055 - loss: 2.4442 - val_accuracy: 0.7316 - v
al_loss: 2.4648 - learning_rate: 0.0010
Epoch 42/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.7874 - loss: 2.4410 - val_accuracy: 0.7096 -
val_loss: 2.4567 - learning_rate: 0.0010
Epoch 43/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8045 - loss: 2.4311 - val_accuracy: 0.7500 - v
al_loss: 2.4444 - learning_rate: 0.0010
Epoch 44/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8024 - loss: 2.4265 - val_accuracy: 0.7353 - v
al_loss: 2.4454 - learning_rate: 0.0010
Epoch 45/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8026 - loss: 2.4217 - val_accuracy: 0.7426 - v
al_loss: 2.4353 - learning_rate: 0.0010
Epoch 46/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8044 - loss: 2.4102 - val_accuracy: 0.7390 - v
al_loss: 2.4319 - learning_rate: 0.0010
Epoch 47/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8011 - loss: 2.4058 - val_accuracy: 0.7353 - v
al_loss: 2.4269 - learning_rate: 0.0010
Epoch 48/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8009 - loss: 2.4018 - val_accuracy: 0.7279 - v
al_loss: 2.4231 - learning_rate: 0.0010
Epoch 49/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8043 - loss: 2.3960 - val_accuracy: 0.7243 - v
al_loss: 2.4181 - learning_rate: 0.0010
Epoch 50/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8171 - loss: 2.3876 - val_accuracy: 0.7390 - v
al_loss: 2.4048 - learning_rate: 0.0010
Epoch 51/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8043 - loss: 2.3803 - val_accuracy: 0.7279 -
val_loss: 2.4045 - learning_rate: 0.0010
Epoch 52/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8246 - loss: 2.3677 - val_accuracy: 0.7316 -
val_loss: 2.3987 - learning_rate: 0.0010
Epoch 53/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8076 - loss: 2.3666 - val_accuracy: 0.7353 -
val_loss: 2.3896 - learning_rate: 0.0010
Epoch 54/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8156 - loss: 2.3583 - val_accuracy: 0.7537 - v
al_loss: 2.3809 - learning_rate: 0.0010
Epoch 55/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8241 - loss: 2.3495 - val_accuracy: 0.7537 -
val_loss: 2.3780 - learning_rate: 0.0010
Epoch 56/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8021 - loss: 2.3515 - val_accuracy: 0.7537 -
val_loss: 2.3688 - learning_rate: 0.0010
Epoch 57/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8341 - loss: 2.3364 - val_accuracy: 0.7463 - v
al_loss: 2.3626 - learning_rate: 0.0010
Epoch 58/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8153 - loss: 2.3345 - val_accuracy: 0.7426 - v
al_loss: 2.3610 - learning_rate: 0.0010
Epoch 59/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8377 - loss: 2.3242 - val_accuracy: 0.7463 -
val_loss: 2.3499 - learning_rate: 0.0010
```

```
Epoch 60/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8248 - loss: 2.3232 - val_accuracy: 0.7721 - v
al_loss: 2.3397 - learning_rate: 0.0010
Epoch 61/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8168 - loss: 2.3178 - val_accuracy: 0.7537 - v
al_loss: 2.3371 - learning_rate: 0.0010
Epoch 62/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8312 - loss: 2.3042 - val_accuracy: 0.7316 -
val_loss: 2.3383 - learning_rate: 0.0010
Epoch 63/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8482 - loss: 2.2980 - val_accuracy: 0.7537 -
val_loss: 2.3244 - learning_rate: 0.0010
Epoch 64/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8353 - loss: 2.2902 - val_accuracy: 0.7500 - v
al_loss: 2.3254 - learning_rate: 0.0010
Epoch 65/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8398 - loss: 2.2855 - val_accuracy: 0.7390 - v
al_loss: 2.3238 - learning_rate: 0.0010
Epoch 66/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8261 - loss: 2.2828 - val_accuracy: 0.7537 - v
al_loss: 2.3105 - learning_rate: 0.0010
Epoch 67/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8417 - loss: 2.2769 - val_accuracy: 0.7537 - v
al_loss: 2.3092 - learning_rate: 0.0010
Epoch 68/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8466 - loss: 2.2688 - val_accuracy: 0.7831 -
val_loss: 2.2999 - learning_rate: 0.0010
Epoch 69/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8413 - loss: 2.2644 - val_accuracy: 0.7463 - v
al_loss: 2.2954 - learning_rate: 0.0010
Epoch 70/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8538 - loss: 2.2525 - val_accuracy: 0.7390 - v
al_loss: 2.2946 - learning_rate: 0.0010
Epoch 71/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8615 - loss: 2.2451 - val_accuracy: 0.7647 - v
al_loss: 2.2771 - learning_rate: 0.0010
Epoch 72/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8343 - loss: 2.2447 - val_accuracy: 0.7721 -
val_loss: 2.2745 - learning_rate: 0.0010
Epoch 73/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8501 - loss: 2.2411 - val_accuracy: 0.7868 - v
al_loss: 2.2585 - learning_rate: 0.0010
Epoch 74/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8506 - loss: 2.2277 - val_accuracy: 0.7721 - v
al_loss: 2.2596 - learning_rate: 0.0010
Epoch 75/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8504 - loss: 2.2242 - val_accuracy: 0.7537 - v
al_loss: 2.2616 - learning_rate: 0.0010
Epoch 76/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8448 - loss: 2.2188 - val_accuracy: 0.7757 - v
al_loss: 2.2517 - learning_rate: 0.0010
Epoch 77/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8631 - loss: 2.2082 - val_accuracy: 0.7757 - v
al_loss: 2.2434 - learning_rate: 0.0010
Epoch 78/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8450 - loss: 2.2089 - val_accuracy: 0.7684 - v
al_loss: 2.2464 - learning_rate: 0.0010
Epoch 79/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8445 - loss: 2.1996 - val_accuracy: 0.7831 - v
```

```
al_loss: 2.2330 - learning_rate: 0.0010
Epoch 80/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8743 - loss: 2.1876 - val_accuracy: 0.7978 -
val_loss: 2.2164 - learning_rate: 0.0010
Epoch 81/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8568 - loss: 2.1870 - val_accuracy: 0.8051 - v
al_loss: 2.2105 - learning_rate: 0.0010
Epoch 82/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8627 - loss: 2.1803 - val_accuracy: 0.7684 - v
al_loss: 2.2109 - learning_rate: 0.0010
Epoch 83/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8522 - loss: 2.1735 - val_accuracy: 0.7647 - v
al_loss: 2.2178 - learning_rate: 0.0010
Epoch 84/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8637 - loss: 2.1670 - val_accuracy: 0.7721 - v
al_loss: 2.2088 - learning_rate: 0.0010
Epoch 85/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8651 - loss: 2.1598 - val_accuracy: 0.7610 - v
al_loss: 2.2005 - learning_rate: 0.0010
Epoch 86/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8559 - loss: 2.1567 - val_accuracy: 0.7610 -
val_loss: 2.2041 - learning_rate: 0.0010
Epoch 87/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8607 - loss: 2.1540 - val_accuracy: 0.7757 - v
al_loss: 2.1967 - learning_rate: 0.0010
Epoch 88/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8682 - loss: 2.1502 - val_accuracy: 0.7941 -
val_loss: 2.1859 - learning_rate: 0.0010
Epoch 89/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8648 - loss: 2.1446 - val_accuracy: 0.7647 -
val_loss: 2.1842 - learning_rate: 0.0010
Epoch 90/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8638 - loss: 2.1325 - val_accuracy: 0.7941 - v
al_loss: 2.1675 - learning_rate: 0.0010
Epoch 91/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8677 - loss: 2.1264 - val_accuracy: 0.7868 - v
al_loss: 2.1664 - learning_rate: 0.0010
Epoch 92/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8602 - loss: 2.1238 - val_accuracy: 0.7941 - v
al_loss: 2.1615 - learning_rate: 0.0010
Epoch 93/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8717 - loss: 2.1182 - val_accuracy: 0.8051 -
val_loss: 2.1538 - learning_rate: 0.0010
Epoch 94/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8670 - loss: 2.1103 - val_accuracy: 0.7868 - v
al_loss: 2.1555 - learning_rate: 0.0010
Epoch 95/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8635 - loss: 2.1056 - val_accuracy: 0.7647 - v
al_loss: 2.1591 - learning_rate: 0.0010
Epoch 96/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8662 - loss: 2.0996 - val_accuracy: 0.7831 - v
al_loss: 2.1434 - learning_rate: 0.0010
Epoch 97/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8938 - loss: 2.0734 - val_accuracy: 0.7721 -
val_loss: 2.1462 - learning_rate: 0.0010
Epoch 98/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8761 - loss: 2.0883 - val_accuracy: 0.7757 -
val_loss: 2.1332 - learning_rate: 0.0010
Epoch 99/150
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8703 - loss: 2.0808 - val_accuracy: 0.7868 - v
al_loss: 2.1289 - learning_rate: 0.0010
Epoch 100/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8677 - loss: 2.0749 - val_accuracy: 0.7831 - v
al_loss: 2.1225 - learning_rate: 0.0010
Epoch 101/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8687 - loss: 2.0716 - val_accuracy: 0.7684 - v
al_loss: 2.1323 - learning_rate: 0.0010
Epoch 102/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8699 - loss: 2.0590 - val_accuracy: 0.7757 -
val_loss: 2.1239 - learning_rate: 0.0010
Epoch 103/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8763 - loss: 2.0583 - val_accuracy: 0.7868 - v
al_loss: 2.1084 - learning_rate: 0.0010
Epoch 104/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8715 - loss: 2.0522 - val_accuracy: 0.7794 - v
al_loss: 2.1056 - learning_rate: 0.0010
Epoch 105/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8881 - loss: 2.0434 - val_accuracy: 0.7831 - v
al_loss: 2.1020 - learning_rate: 0.0010
Epoch 106/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8805 - loss: 2.0390 - val_accuracy: 0.8015 - v
al_loss: 2.0948 - learning_rate: 0.0010
Epoch 107/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8758 - loss: 2.0350 - val_accuracy: 0.7941 -
val_loss: 2.0963 - learning_rate: 0.0010
Epoch 108/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8902 - loss: 2.0224 - val_accuracy: 0.7941 -
val_loss: 2.0780 - learning_rate: 0.0010
Epoch 109/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8799 - loss: 2.0150 - val_accuracy: 0.7831 -
val_loss: 2.0783 - learning_rate: 0.0010
Epoch 110/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8618 - loss: 2.0234 - val_accuracy: 0.7978 -
val_loss: 2.0822 - learning_rate: 0.0010
Epoch 111/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8682 - loss: 2.0142 - val_accuracy: 0.7684 - v
al_loss: 2.0775 - learning_rate: 0.0010
Epoch 112/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8819 - loss: 2.0012 - val_accuracy: 0.7941 -
val_loss: 2.0761 - learning_rate: 0.0010
Epoch 113/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 11s/step - accuracy: 0.8788 - loss: 2.0007 - val_accuracy: 0.7941 -
val_loss: 2.0689 - learning_rate: 0.0010
Epoch 114/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8840 - loss: 1.9984 - val_accuracy: 0.7831 - v
al_loss: 2.0597 - learning_rate: 0.0010
Epoch 115/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8715 - loss: 1.9964 - val_accuracy: 0.7868 -
val_loss: 2.0476 - learning_rate: 0.0010
Epoch 116/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8686 - loss: 1.9882 - val_accuracy: 0.8088 -
val_loss: 2.0416 - learning_rate: 0.0010
Epoch 117/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8742 - loss: 1.9819 - val_accuracy: 0.7831 - v
al_loss: 2.0529 - learning_rate: 0.0010
Epoch 118/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8855 - loss: 1.9755 - val_accuracy: 0.7904 - v
al_loss: 2.0323 - learning_rate: 0.0010
```

```
Epoch 119/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8828 - loss: 1.9713 - val_accuracy: 0.7978 - v
al_loss: 2.0322 - learning_rate: 0.0010
Epoch 120/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8762 - loss: 1.9674 - val_accuracy: 0.7904 - v
al_loss: 2.0263 - learning_rate: 0.0010
Epoch 121/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.9080 - loss: 1.9522 - val_accuracy: 0.7978 -
val_loss: 2.0216 - learning_rate: 0.0010
Epoch 122/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.9003 - loss: 1.9501 - val_accuracy: 0.8051 -
val_loss: 2.0143 - learning_rate: 0.0010
Epoch 123/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8793 - loss: 1.9573 - val_accuracy: 0.7941 -
val_loss: 2.0103 - learning_rate: 0.0010
Epoch 124/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8862 - loss: 1.9498 - val_accuracy: 0.8162 -
val_loss: 1.9856 - learning_rate: 0.0010
Epoch 125/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8792 - loss: 1.9424 - val_accuracy: 0.7904 - v
al_loss: 2.0049 - learning_rate: 0.0010
Epoch 126/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8845 - loss: 1.9326 - val_accuracy: 0.8088 - v
al_loss: 1.9944 - learning_rate: 0.0010
Epoch 127/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.8899 - loss: 1.9266
Epoch 127: ReduceLROnPlateau reducing learning rate to 0.0006000000284984708.
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8901 - loss: 1.9260 - val_accuracy: 0.8235 - v
al_loss: 1.9868 - learning_rate: 0.0010
Epoch 128/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8788 - loss: 1.9338 - val_accuracy: 0.7978 - v
al_loss: 1.9842 - learning_rate: 6.0000e-04
Epoch 129/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8821 - loss: 1.9233 - val_accuracy: 0.7831 -
val_loss: 2.0065 - learning_rate: 6.0000e-04
Epoch 130/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8814 - loss: 1.9125 - val_accuracy: 0.8125 -
val_loss: 1.9864 - learning_rate: 6.0000e-04
Epoch 131/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8924 - loss: 1.9119 - val_accuracy: 0.8125 - v
al_loss: 1.9733 - learning_rate: 6.0000e-04
Epoch 132/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8935 - loss: 1.9091 - val_accuracy: 0.7941 - v
al_loss: 1.9793 - learning_rate: 6.0000e-04
Epoch 133/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8860 - loss: 1.9080 - val_accuracy: 0.8015 - v
al_loss: 1.9726 - learning_rate: 6.0000e-04
Epoch 134/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8926 - loss: 1.9018 - val_accuracy: 0.8015 - v
al_loss: 1.9679 - learning_rate: 6.0000e-04
Epoch 135/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.8897 - loss: 1.9035 - val_accuracy: 0.8015 -
val_loss: 1.9636 - learning_rate: 6.0000e-04
Epoch 136/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8923 - loss: 1.8998 - val_accuracy: 0.8235 - v
al_loss: 1.9583 - learning_rate: 6.0000e-04
Epoch 137/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8947 - loss: 1.8889 - val_accuracy: 0.7941 - v
al_loss: 1.9669 - learning_rate: 6.0000e-04
```

```
Epoch 138/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8942 - loss: 1.8921 - val_accuracy: 0.8162 - v
al_loss: 1.9526 - learning_rate: 6.0000e-04
Epoch 139/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8951 - loss: 1.8854 - val_accuracy: 0.8051 - v
al_loss: 1.9569 - learning_rate: 6.0000e-04
Epoch 140/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8967 - loss: 1.8822 - val_accuracy: 0.8125 - v
al_loss: 1.9565 - learning_rate: 6.0000e-04
Epoch 141/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.9105 - loss: 1.8707
Epoch 141: ReduceLROnPlateau reducing learning rate to 0.0003600000170990825.
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.9058 - loss: 1.8734 - val_accuracy: 0.8015 -
val_loss: 1.9628 - learning_rate: 6.0000e-04
Epoch 142/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.9108 - loss: 1.8669 - val_accuracy: 0.8088 -
val_loss: 1.9443 - learning_rate: 3.6000e-04
Epoch 143/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.9008 - loss: 1.8747 - val_accuracy: 0.8088 - v
al_loss: 1.9447 - learning_rate: 3.6000e-04
Epoch 144/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8938 - loss: 1.8802 - val_accuracy: 0.7941 - v
al_loss: 1.9466 - learning_rate: 3.6000e-04
Epoch 145/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.8948 - loss: 1.8684
Epoch 145: ReduceLROnPlateau reducing learning rate to 0.00021600000327453016.
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8944 - loss: 1.8687 - val_accuracy: 0.7978 - v
al_loss: 1.9445 - learning_rate: 3.6000e-04
Epoch 146/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 8s/step - accuracy: 0.8879 - loss: 1.8734 - val_accuracy: 0.7978 - v
al_loss: 1.9456 - learning_rate: 2.1600e-04
Epoch 147/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8989 - loss: 1.8695 - val_accuracy: 0.8088 - v
al_loss: 1.9397 - learning_rate: 2.1600e-04
Epoch 148/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 11s/step - accuracy: 0.9036 - loss: 1.8606 - val_accuracy: 0.8235 -
val_loss: 1.9360 - learning_rate: 2.1600e-04
Epoch 149/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8958 - loss: 1.8682 - val_accuracy: 0.8015 - v
al_loss: 1.9437 - learning_rate: 2.1600e-04
Epoch 150/150
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8940 - loss: 1.8619 - val_accuracy: 0.8015 - v
al_loss: 1.9442 - learning_rate: 2.1600e-04
```

In [28]:

```python
#check metrics on test data and plot accuracy curve

animal_acc_tx_1 = history_tx_1.history['accuracy']
animal_val_tx_1 = history_tx_1.history['val_accuracy']

epochs_ind = [i for i in range(1, 1 + epochs)]

plt.plot(epochs_ind, animal_acc_tx_1, 'blue', label = 'Training accuracy')
plt.plot(epochs_ind, animal_val_tx_1, 'red', label = 'Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy over training epochs')
plt.legend()
plt.show()
```

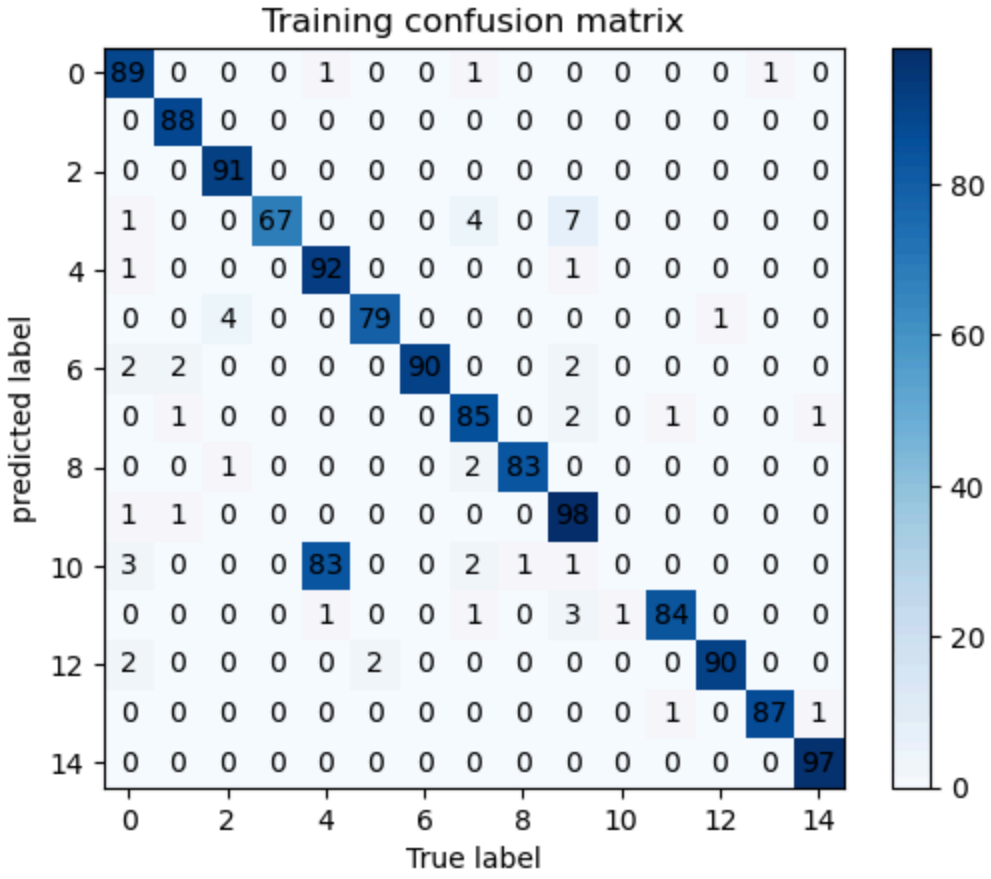
In [29]:
```
_, train_acc = animal_tx_1.evaluate(X_train, Y_train, verbose = 0)
_, test_acc = animal_tx_1.evaluate(X_test, Y_test, verbose = 0)

print(f"Training accuracy: {train_acc*100}%")
print(f"Testing accuracy: {test_acc*100}%")
```

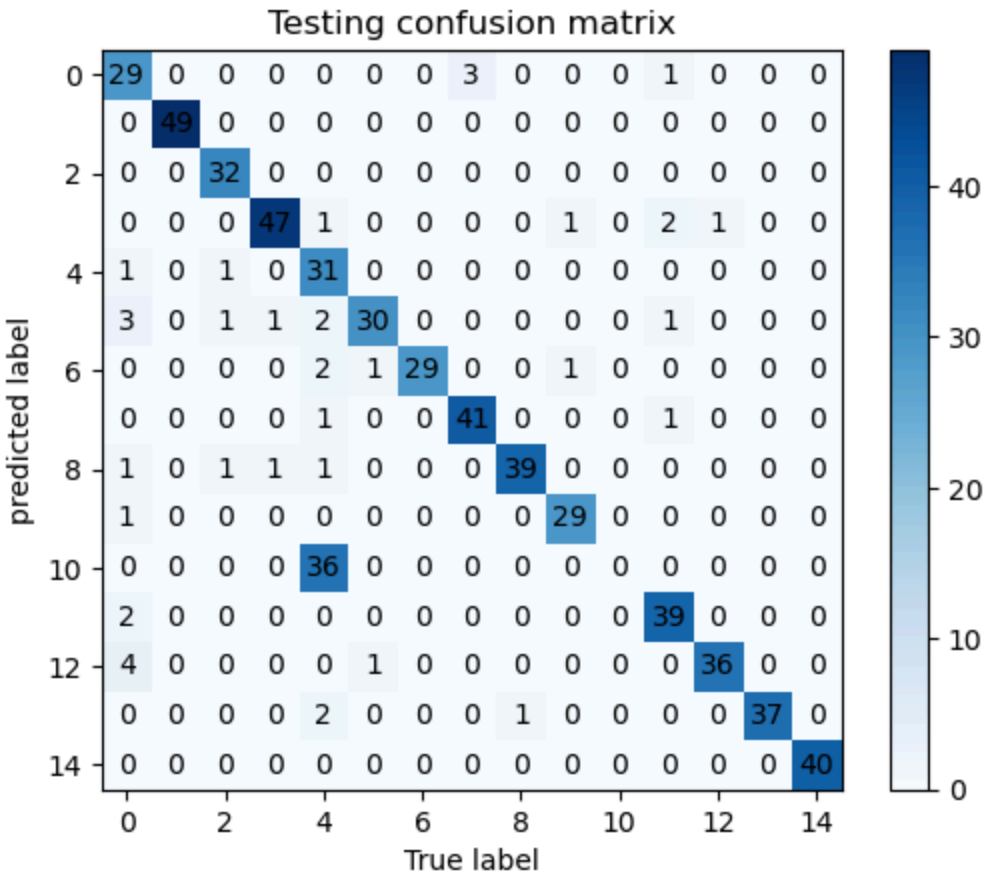
```
Training accuracy: 89.70588445663452%
Testing accuracy: 86.98630332946777%
```

In [30]:
```
#confusion matrix

train_preds = np.argmax(animal_tx_1.predict(X_train), axis = 1)
test_preds = np.argmax(animal_tx_1.predict(X_test), axis = 1)

train_cm = confusion_matrix(np.argmax(Y_train, axis = 1), train_preds)
test_cm = confusion_matrix(np.argmax(Y_test, axis = 1), test_preds)

plt.imshow(train_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(train_cm.shape[0]):
  for j in range(train_cm.shape[1]):
    plt.text(j, i, train_cm[i, j], ha = 'center', va = 'center')
plt.title('Training confusion matrix')
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()

plt.imshow(test_cm, cmap = plt.cm.Blues)
plt.colorbar()
for i in range(test_cm.shape[0]):
  for j in range(test_cm.shape[1]):
    plt.text(j, i, test_cm[i, j], ha = 'center', va = 'center')
```

```
plt.title('Testing confusion matrix')
plt.xlabel('True label')
plt.ylabel('predicted label')
plt.show()
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 182ms/step
19/19 ━━━━━━━━━━━━━━━━━━━━ 3s 174ms/step
```





This is very close but not quite what I want. I will now add some layers that may help with generalization, as well as unfreezing a few layers in the MobileNetV2 model to allow accuracy to climb higher. Last round also took significant time to lower the learning rate, so I will start at an even higher learning rate to help it converge quickly. I am additionally adding more maximum epochs and early stopping to allow the model to achieve higher train accuracy, but still prevent overfitting if the model plateaus for an extended period.

In [43]:
```
temp_tx_2 = MobileNetV2(weights = 'imagenet', include_top = False, input_shape = X_shape[1:])

#trainaing the last 10 layers in the transferred model
temp_tx_2.trainable = True
for layer in temp_tx_2.layers[:-10]:
    layer.trainable = False


animal_tx_2 = Sequential()
animal_tx_2.add(temp_tx_2)

animal_tx_2.add(GlobalAveragePooling2D())

#adding another layer with regularization
animal_tx_2.add(Dense(248,
                      activation = None,
                     kernel_regularizer = regularizers.l2(0.0005)))
animal_tx_2.add(BatchNormalization())
animal_tx_2.add(Activation('relu'))

#adding some more generalization to balance the extra training
animal_tx_2.add(Dropout(0.5))

#same output as in first
```

```
animal_tx_2.add(Dense(Y_shape[1], activation = 'softmax'))

#Last round took a long time to reduce learning rate, let's start higher
optimizer = Adam(learning_rate = 0.002, amsgrad = True)

animal_tx_2.compile(loss = 'categorical_crossentropy',
                    optimizer = optimizer,
                    metrics = ['accuracy'])

animal_tx_2.summary()
```

Model: "sequential_11"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_6 (GlobalAveragePooling2D)	(None, 1280)	0
dense_11 (Dense)	(None, 248)	317,688
batch_normalization_1 (BatchNormalization)	(None, 248)	992
activation_1 (Activation)	(None, 248)	0
dropout_5 (Dropout)	(None, 248)	0
dense_12 (Dense)	(None, 15)	3,735

Total params: 2,580,399 (9.84 MB)

Trainable params: 1,054,399 (4.02 MB)

Non-trainable params: 1,526,000 (5.82 MB)

In [44]:

```
#reduces learning rate if loss value plateaus for 3 iterations
reduce_2 = ReduceLROnPlateau(
    monitor = 'val_loss',
    patience = 3,
    verbose = 1,
    factor = 0.6,
    min_lr = 0.000005
)

#early stopping to prevent some overfitting
early_stop = EarlyStopping(monitor = 'val_loss',
                           patience = 8, #higher than the LR reduction
                           restore_best_weights = True)

#train the model
epochs = 200 #More epochs since early stopping should prevent reaching if overfitting starts
batch_size = 32

history_tx_2 = animal_tx_2.fit(train_gen,
                               epochs = epochs,
                               batch_size = batch_size,
                              validation_data = val_gen,
                             callbacks = [reduce_2])
```

```
Epoch 1/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 43s 15s/step - accuracy: 0.2405 - loss: 2.9058 - val_accuracy: 0.4816 -
val_loss: 1.9326 - learning_rate: 0.0020
Epoch 2/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.6111 - loss: 1.4699 - val_accuracy: 0.6066 -
val_loss: 1.7406 - learning_rate: 0.0020
Epoch 3/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.7409 - loss: 1.0882 - val_accuracy: 0.5882 - v
al_loss: 2.3190 - learning_rate: 0.0020
Epoch 4/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.7840 - loss: 0.9294 - val_accuracy: 0.5956 - v
al_loss: 2.7935 - learning_rate: 0.0020
Epoch 5/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.8080 - loss: 0.8273
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0012000000569969416.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8098 - loss: 0.8261 - val_accuracy: 0.5919 - v
al_loss: 3.1869 - learning_rate: 0.0020
Epoch 6/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.8336 - loss: 0.7953 - val_accuracy: 0.5772 -
val_loss: 3.3817 - learning_rate: 0.0012
Epoch 7/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.8641 - loss: 0.7002 - val_accuracy: 0.5735 - v
al_loss: 3.3530 - learning_rate: 0.0012
Epoch 8/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.8581 - loss: 0.6832
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.000720000034198165.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.8586 - loss: 0.6764 - val_accuracy: 0.5404 -
val_loss: 3.6357 - learning_rate: 0.0012
Epoch 9/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8782 - loss: 0.6226 - val_accuracy: 0.5221 - v
al_loss: 4.0387 - learning_rate: 7.2000e-04
Epoch 10/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.8743 - loss: 0.6446 - val_accuracy: 0.4706 - v
al_loss: 4.3310 - learning_rate: 7.2000e-04
Epoch 11/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.8871 - loss: 0.6128
Epoch 11: ReduceLROnPlateau reducing learning rate to 0.0004320000065490603.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.8898 - loss: 0.6050 - val_accuracy: 0.4853 -
val_loss: 4.4287 - learning_rate: 7.2000e-04
Epoch 12/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9063 - loss: 0.5614 - val_accuracy: 0.4743 - v
al_loss: 4.5064 - learning_rate: 4.3200e-04
Epoch 13/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.8934 - loss: 0.5896 - val_accuracy: 0.4412 -
val_loss: 4.9562 - learning_rate: 4.3200e-04
Epoch 14/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9222 - loss: 0.5147
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.00025920000043697653.
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9201 - loss: 0.5187 - val_accuracy: 0.4485 - v
al_loss: 5.1103 - learning_rate: 4.3200e-04
Epoch 15/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9113 - loss: 0.5192 - val_accuracy: 0.3971 - v
al_loss: 5.3986 - learning_rate: 2.5920e-04
Epoch 16/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9105 - loss: 0.5181 - val_accuracy: 0.3676 - v
al_loss: 5.3832 - learning_rate: 2.5920e-04
Epoch 17/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.8957 - loss: 0.5488
Epoch 17: ReduceLROnPlateau reducing learning rate to 0.00015551999676972626.
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9013 - loss: 0.5365 - val_accuracy: 0.4265 -
val_loss: 5.3309 - learning_rate: 2.5920e-04
Epoch 18/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9196 - loss: 0.5009 - val_accuracy: 0.4154 -
val_loss: 5.3256 - learning_rate: 1.5552e-04
Epoch 19/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9248 - loss: 0.4606 - val_accuracy: 0.4007 -
val_loss: 5.4468 - learning_rate: 1.5552e-04
Epoch 20/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.9236 - loss: 0.4706
Epoch 20: ReduceLROnPlateau reducing learning rate to 9.331199980806559e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9250 - loss: 0.4733 - val_accuracy: 0.4338 -
val_loss: 5.3988 - learning_rate: 1.5552e-04
Epoch 21/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9304 - loss: 0.4890 - val_accuracy: 0.3640 - v
al_loss: 5.3752 - learning_rate: 9.3312e-05
Epoch 22/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9345 - loss: 0.4616 - val_accuracy: 0.3493 - v
al_loss: 5.8677 - learning_rate: 9.3312e-05
Epoch 23/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9202 - loss: 0.4726
Epoch 23: ReduceLROnPlateau reducing learning rate to 5.598720163106918e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9206 - loss: 0.4729 - val_accuracy: 0.4081 - v
al_loss: 5.8098 - learning_rate: 9.3312e-05
Epoch 24/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9301 - loss: 0.4835 - val_accuracy: 0.3787 - v
al_loss: 5.8907 - learning_rate: 5.5987e-05
Epoch 25/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9250 - loss: 0.4695 - val_accuracy: 0.3787 -
val_loss: 6.0276 - learning_rate: 5.5987e-05
Epoch 26/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9311 - loss: 0.4768
Epoch 26: ReduceLROnPlateau reducing learning rate to 3.359232141519897e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9315 - loss: 0.4752 - val_accuracy: 0.3640 - v
al_loss: 5.6801 - learning_rate: 5.5987e-05
Epoch 27/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9273 - loss: 0.4585 - val_accuracy: 0.3713 -
val_loss: 5.7638 - learning_rate: 3.3592e-05
Epoch 28/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 7s/step - accuracy: 0.9305 - loss: 0.4613 - val_accuracy: 0.3713 - v
al_loss: 5.6620 - learning_rate: 3.3592e-05
Epoch 29/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9167 - loss: 0.5001
Epoch 29: ReduceLROnPlateau reducing learning rate to 2.015539284911938e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9208 - loss: 0.4914 - val_accuracy: 0.3603 - v
al_loss: 5.7105 - learning_rate: 3.3592e-05
Epoch 30/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9370 - loss: 0.4590 - val_accuracy: 0.3824 - v
al_loss: 5.6360 - learning_rate: 2.0155e-05
Epoch 31/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9379 - loss: 0.4562 - val_accuracy: 0.3713 - v
al_loss: 5.8559 - learning_rate: 2.0155e-05
Epoch 32/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 9s/step - accuracy: 0.9394 - loss: 0.4475
Epoch 32: ReduceLROnPlateau reducing learning rate to 1.209323527291417e-05.
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9378 - loss: 0.4492 - val_accuracy: 0.3787 -
val_loss: 5.7970 - learning_rate: 2.0155e-05
Epoch 33/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 12s/step - accuracy: 0.9450 - loss: 0.4455 - val_accuracy: 0.3309 -
```

```
val_loss: 5.8888 - learning_rate: 1.2093e-05
Epoch 34/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9316 - loss: 0.4901 - val_accuracy: 0.3529 - v
al_loss: 6.0274 - learning_rate: 1.2093e-05
Epoch 35/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9235 - loss: 0.4778
Epoch 35: ReduceLROnPlateau reducing learning rate to 7.255941272887867e-06.
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9238 - loss: 0.4791 - val_accuracy: 0.3676 - v
al_loss: 5.8412 - learning_rate: 1.2093e-05
Epoch 36/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9320 - loss: 0.4668 - val_accuracy: 0.3713 - v
al_loss: 5.7893 - learning_rate: 7.2559e-06
Epoch 37/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9389 - loss: 0.4391 - val_accuracy: 0.3640 - v
al_loss: 5.5635 - learning_rate: 7.2559e-06
Epoch 38/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9408 - loss: 0.4440
Epoch 38: ReduceLROnPlateau reducing learning rate to 5e-06.
3/3 ━━━━━━━━━━━━━━━━━━━━ 24s 7s/step - accuracy: 0.9404 - loss: 0.4444 - val_accuracy: 0.3750 - v
al_loss: 5.7830 - learning_rate: 7.2559e-06
Epoch 39/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9342 - loss: 0.4625 - val_accuracy: 0.3529 - v
al_loss: 5.7866 - learning_rate: 5.0000e-06
Epoch 40/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9404 - loss: 0.4518 - val_accuracy: 0.3640 - v
al_loss: 5.3814 - learning_rate: 5.0000e-06
Epoch 41/200
3/3 ━━━━━━━━━━━━━━━━━━━━ 25s 8s/step - accuracy: 0.9334 - loss: 0.4715 - val_accuracy: 0.4044 - v
al_loss: 5.4993 - learning_rate: 5.0000e-06
Epoch 42/200
2/3 ━━━━━━━━━━━━──────── 9s 9s/step - accuracy: 0.9220 - loss: 0.4887
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[44], line 19
     16 epochs = 200 #More epochs since early stopping should prevent reaching if overfitting sta
rts
     17 batch_size = 32
---> 19 history_tx_2 = animal_tx_2.fit(train_gen,
     20                                  epochs = epochs,
     21                                  batch_size = batch_size,
     22                                 validation_data = val_gen,
     23                                callbacks = [reduce_2])

File ~\AppData\Roaming\Python\Python312\site-packages\keras\src\utils\traceback_utils.py:117, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\AppData\Roaming\Python\Python312\site-packages\keras\src\backend\tensorflow\trainer.py:37
1, in TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split,
validation_data, shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, validation
_steps, validation_batch_size, validation_freq)
    369 for step, iterator in epoch_iterator:
    370     callbacks.on_train_batch_begin(step)
--> 371     logs = self.train_function(iterator)
    372     callbacks.on_train_batch_end(step, logs)
    373     if self.stop_training:

File ~\AppData\Roaming\Python\Python312\site-packages\keras\src\backend\tensorflow\trainer.py:21
9, in TensorFlowTrainer._make_function.<locals>.function(iterator)
    215 def function(iterator):
    216     if isinstance(
    217         iterator, (tf.data.Iterator, tf.distribute.DistributedIterator)
    218     ):
--> 219         opt_outputs = multi_step_on_iterator(iterator)
    220         if not opt_outputs.has_value():
    221             raise StopIteration

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\util\traceback_utils.py:1
50, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_functio
n\polymorphic_function.py:833, in Function.__call__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_functio
n\polymorphic_function.py:878, in Function._call(self, *args, **kwds)
    875 self._lock.release()
```

```
    876 # In this case we have not created variables on the first call. So we can
    877 # run the first trace but we should fail if variables are created.
--> 878 results = tracing_compilation.call_function(
    879     args, kwds, self._variable_creation_config
    880 )
    881 if self._created_variables:
    882   raise ValueError("Creating variables on a non-first call to a function"
    883                    " decorated with tf.function.")

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_functio
n\tracing_compilation.py:139, in call_function(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_functio
n\concrete_function.py:1322, in ConcreteFunction._call_flat(self, tensor_inputs, captured_inputs)
   1318 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1319 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1320     and executing_eagerly):
   1321   # No tape is watching; skip to running the function.
-> 1322   return self._inference_function.call_preflattened(args)
   1323 forward_backward = self._select_forward_and_backward_functions(
   1324     args,
   1325     possible_gradient_type,
   1326     executing_eagerly)
   1327 forward_function, args_with_tangents = forward_backward.forward()

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_functio
n\atomic_function.py:216, in AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) -> Any:
    215   """Calls with flattened tensor inputs and returns the structured output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\polymorphic_functio
n\atomic_function.py:251, in AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\context.py:1688, in
Context.call_function(self, name, tensor_inputs, num_outputs)
   1686 cancellation_context = cancellation.context()
   1687 if cancellation_context is None:
-> 1688   outputs = execute.execute(
   1689       name.decode("utf-8"),
```

```
  1690       num_outputs=num_outputs,
  1691       inputs=tensor_inputs,
  1692       attrs=attrs,
  1693       ctx=self,
  1694   )
  1695 else:
  1696   outputs = execute.execute_with_cancellation(
  1697       name.decode("utf-8"),
  1698       num_outputs=num_outputs,
  (...)
  1702       cancellation_manager=cancellation_context,
  1703   )

File ~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\execute.py:53, in q
uick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

I stopped this early due to obvious overfitting. I will return to the model above, but introduce more transformations to the image generator in an attempt to allow it to generalize even better, and I will maintain the higher starting learning rate and increased number of epochs. I will still be using one additional Dense layer as I believe the above model could do better in ending accuracy. If this does not work I will remove it and just allow the model to run for longer.

In [18]:
```
generator2 = ImageDataGenerator(
    rotation_range = 50,
    width_shift_range = 0.15,
    height_shift_range = 0.15,
    horizontal_flip = True,
    vertical_flip = True,
    zoom_range = 0.11,
    validation_split = 0.2
)

train_gen_2 = generator2.flow(X_train, Y_train, subset = "training", batch_size = 1000)
val_gen_2 = generator2.flow(X_train, Y_train, subset = "validation", batch_size = 1000)
```

In [19]:
```
temp_tx = MobileNetV2(weights = 'imagenet', include_top = False, input_shape = X_shape[1:])

for layer in temp_tx.layers:
    layer.trainable = False

temp_tx.layers[-1].trainable = True

animal_tx_3 = Sequential()
animal_tx_3.add(temp_tx)

animal_tx_3.add(GlobalAveragePooling2D())

animal_tx_3.add(Dense(248,
                      activation = None,
                      kernel_regularizer = regularizers.l1_l2(0.001, 0.0005)))
```

```
animal_tx_3.add(BatchNormalization())
animal_tx_3.add(Activation('relu'))
animal_tx_3.add(Dropout(0.5))

animal_tx_3.add(Dense(Y_shape[1], activation = 'softmax'))

optimizer = Adam(learning_rate = 0.003, amsgrad = True)

animal_tx_3.compile(loss = 'categorical_crossentropy',
                    optimizer = optimizer,
                    metrics = ['accuracy'])

animal_tx_3.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 248)	317,688
batch_normalization (BatchNormalization)	(None, 248)	992
activation (Activation)	(None, 248)	0
dropout (Dropout)	(None, 248)	0
dense_1 (Dense)	(None, 15)	3,735

Total params: 2,580,399 (9.84 MB)

Trainable params: 321,919 (1.23 MB)

Non-trainable params: 2,258,480 (8.62 MB)

In [20]:
```
#reduces learning rate if loss value plateaus for 3 iterations
reduce_2 = ReduceLROnPlateau(
    monitor = 'val_loss',
    patience = 3,
    verbose = 1,
    factor = 0.6,
    min_lr = 0.000005
)

#early stopping to prevent some overfitting
early_stop = EarlyStopping(monitor = 'val_loss',
                           patience = 10, #higher than the LR reduction
                           restore_best_weights = True)

#train the model
epochs = 250 #More epochs since early stopping should prevent reaching if overfitting starts
batch_size = 32

history_tx_3 = animal_tx_3.fit(train_gen_2,
                               epochs = epochs,
```

```
                                    batch_size = batch_size,
                                   validation_data = val_gen_2,
                                  callbacks = [reduce_2, early_stop])
```

```
C:\Users\lcleymaet\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 52s 41s/step - accuracy: 0.1176 - loss: 13.2799 - val_accuracy: 0.2904 -
val_loss: 12.0832 - learning_rate: 0.0030
Epoch 2/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.4631 - loss: 11.5920 - val_accuracy: 0.4118 -
val_loss: 11.4353 - learning_rate: 0.0030
Epoch 3/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.6271 - loss: 10.6860 - val_accuracy: 0.4963 -
val_loss: 10.8457 - learning_rate: 0.0030
Epoch 4/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.7071 - loss: 10.1737 - val_accuracy: 0.5846 -
val_loss: 10.1741 - learning_rate: 0.0030
Epoch 5/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.7265 - loss: 9.5137 - val_accuracy: 0.6360 -
val_loss: 9.6013 - learning_rate: 0.0030
Epoch 6/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.7660 - loss: 9.1293 - val_accuracy: 0.6213 - v
al_loss: 9.1652 - learning_rate: 0.0030
Epoch 7/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.7964 - loss: 8.5038 - val_accuracy: 0.6507 -
val_loss: 8.6692 - learning_rate: 0.0030
Epoch 8/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.7783 - loss: 8.0410 - val_accuracy: 0.6250 -
val_loss: 8.2745 - learning_rate: 0.0030
Epoch 9/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8117 - loss: 7.6487 - val_accuracy: 0.6654 - v
al_loss: 7.7172 - learning_rate: 0.0030
Epoch 10/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8166 - loss: 7.1730 - val_accuracy: 0.6654 - v
al_loss: 7.2455 - learning_rate: 0.0030
Epoch 11/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8165 - loss: 6.6146 - val_accuracy: 0.7059 -
val_loss: 6.7300 - learning_rate: 0.0030
Epoch 12/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8571 - loss: 6.2148 - val_accuracy: 0.7059 - v
al_loss: 6.2471 - learning_rate: 0.0030
Epoch 13/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8737 - loss: 5.7622 - val_accuracy: 0.7353 - v
al_loss: 5.8215 - learning_rate: 0.0030
Epoch 14/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8557 - loss: 5.2784 - val_accuracy: 0.7426 -
val_loss: 5.4029 - learning_rate: 0.0030
Epoch 15/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8732 - loss: 4.8878 - val_accuracy: 0.7279 -
val_loss: 5.0611 - learning_rate: 0.0030
Epoch 16/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8612 - loss: 4.5708 - val_accuracy: 0.7390 -
val_loss: 4.6894 - learning_rate: 0.0030
Epoch 17/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8797 - loss: 4.3015 - val_accuracy: 0.7610 - v
al_loss: 4.3868 - learning_rate: 0.0030
Epoch 18/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8600 - loss: 3.9873 - val_accuracy: 0.7941 -
val_loss: 4.0705 - learning_rate: 0.0030
Epoch 19/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8612 - loss: 3.7094 - val_accuracy: 0.7868 -
val_loss: 3.8543 - learning_rate: 0.0030
Epoch 20/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8725 - loss: 3.4786 - val_accuracy: 0.7794 -
```

```
val_loss: 3.6511 - learning_rate: 0.0030
Epoch 21/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8905 - loss: 3.2257 - val_accuracy: 0.7537 -
val_loss: 3.5099 - learning_rate: 0.0030
Epoch 22/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8775 - loss: 3.0933 - val_accuracy: 0.7941 - v
al_loss: 3.3335 - learning_rate: 0.0030
Epoch 23/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8642 - loss: 2.9594 - val_accuracy: 0.7794 -
val_loss: 3.2300 - learning_rate: 0.0030
Epoch 24/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8549 - loss: 2.8634 - val_accuracy: 0.7794 -
val_loss: 3.1322 - learning_rate: 0.0030
Epoch 25/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8722 - loss: 2.7337 - val_accuracy: 0.7610 - v
al_loss: 3.0986 - learning_rate: 0.0030
Epoch 26/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8713 - loss: 2.6354 - val_accuracy: 0.7426 - v
al_loss: 3.0240 - learning_rate: 0.0030
Epoch 27/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8540 - loss: 2.5615 - val_accuracy: 0.7463 - v
al_loss: 3.0569 - learning_rate: 0.0030
Epoch 28/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8782 - loss: 2.4571 - val_accuracy: 0.7500 - v
al_loss: 2.9774 - learning_rate: 0.0030
Epoch 29/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8655 - loss: 2.4176 - val_accuracy: 0.7279 - v
al_loss: 2.9785 - learning_rate: 0.0030
Epoch 30/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8709 - loss: 2.4022 - val_accuracy: 0.7500 - v
al_loss: 2.9381 - learning_rate: 0.0030
Epoch 31/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8615 - loss: 2.3604 - val_accuracy: 0.7647 - v
al_loss: 2.9446 - learning_rate: 0.0030
Epoch 32/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8738 - loss: 2.3483 - val_accuracy: 0.7353 - v
al_loss: 2.9258 - learning_rate: 0.0030
Epoch 33/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8449 - loss: 2.3876 - val_accuracy: 0.7353 - v
al_loss: 2.8656 - learning_rate: 0.0030
Epoch 34/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8396 - loss: 2.4218 - val_accuracy: 0.7059 -
val_loss: 2.9639 - learning_rate: 0.0030
Epoch 35/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.8225 - loss: 2.5041 - val_accuracy: 0.6985 -
val_loss: 3.0072 - learning_rate: 0.0030
Epoch 36/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 446ms/step - accuracy: 0.8367 - loss: 2.4871
Epoch 36: ReduceLROnPlateau reducing learning rate to 0.0018000000015646219.
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8369 - loss: 2.4873 - val_accuracy: 0.7096 - v
al_loss: 2.9924 - learning_rate: 0.0030
Epoch 37/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8498 - loss: 2.4750 - val_accuracy: 0.7316 - v
al_loss: 2.9442 - learning_rate: 0.0018
Epoch 38/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8572 - loss: 2.4229 - val_accuracy: 0.7316 -
val_loss: 2.9224 - learning_rate: 0.0018
Epoch 39/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8624 - loss: 2.4325 - val_accuracy: 0.7574 -
```

```
val_loss: 2.8289 - learning_rate: 0.0018
Epoch 40/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8505 - loss: 2.3708 - val_accuracy: 0.7757 -
val_loss: 2.7664 - learning_rate: 0.0018
Epoch 41/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8712 - loss: 2.2874 - val_accuracy: 0.7647 - v
al_loss: 2.7274 - learning_rate: 0.0018
Epoch 42/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8866 - loss: 2.2143 - val_accuracy: 0.8051 - v
al_loss: 2.6070 - learning_rate: 0.0018
Epoch 43/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8880 - loss: 2.1645 - val_accuracy: 0.8088 - v
al_loss: 2.5531 - learning_rate: 0.0018
Epoch 44/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9026 - loss: 2.0556 - val_accuracy: 0.7721 -
val_loss: 2.5721 - learning_rate: 0.0018
Epoch 45/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8901 - loss: 2.0338 - val_accuracy: 0.7463 - v
al_loss: 2.5494 - learning_rate: 0.0018
Epoch 46/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8703 - loss: 2.0146 - val_accuracy: 0.7941 - v
al_loss: 2.4690 - learning_rate: 0.0018
Epoch 47/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8951 - loss: 1.9289 - val_accuracy: 0.7537 - v
al_loss: 2.4344 - learning_rate: 0.0018
Epoch 48/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8987 - loss: 1.8650 - val_accuracy: 0.7794 -
val_loss: 2.4135 - learning_rate: 0.0018
Epoch 49/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8899 - loss: 1.8812 - val_accuracy: 0.8162 - v
al_loss: 2.3275 - learning_rate: 0.0018
Epoch 50/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9008 - loss: 1.8413 - val_accuracy: 0.7904 - v
al_loss: 2.3569 - learning_rate: 0.0018
Epoch 51/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8863 - loss: 1.8019 - val_accuracy: 0.7904 -
val_loss: 2.3326 - learning_rate: 0.0018
Epoch 52/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 444ms/step - accuracy: 0.8869 - loss: 1.8579
Epoch 52: ReduceLROnPlateau reducing learning rate to 0.0010799999814480542.
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8875 - loss: 1.8563 - val_accuracy: 0.7647 - v
al_loss: 2.3371 - learning_rate: 0.0018
Epoch 53/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8807 - loss: 1.8564 - val_accuracy: 0.7610 - v
al_loss: 2.3224 - learning_rate: 0.0011
Epoch 54/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8822 - loss: 1.7876 - val_accuracy: 0.8051 - v
al_loss: 2.2095 - learning_rate: 0.0011
Epoch 55/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8942 - loss: 1.7413 - val_accuracy: 0.8088 - v
al_loss: 2.1921 - learning_rate: 0.0011
Epoch 56/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.8983 - loss: 1.7223 - val_accuracy: 0.8235 - v
al_loss: 2.2038 - learning_rate: 0.0011
Epoch 57/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9055 - loss: 1.6509 - val_accuracy: 0.8125 -
val_loss: 2.1337 - learning_rate: 0.0011
Epoch 58/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8890 - loss: 1.6421 - val_accuracy: 0.8125 -
```

```
val_loss: 2.1263 - learning_rate: 0.0011
Epoch 59/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9168 - loss: 1.5852 - val_accuracy: 0.8235 -
val_loss: 2.1134 - learning_rate: 0.0011
Epoch 60/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9160 - loss: 1.6118 - val_accuracy: 0.8088 -
val_loss: 2.0947 - learning_rate: 0.0011
Epoch 61/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9235 - loss: 1.5321 - val_accuracy: 0.8125 -
val_loss: 2.0671 - learning_rate: 0.0011
Epoch 62/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9022 - loss: 1.5573 - val_accuracy: 0.7794 - v
al_loss: 2.0714 - learning_rate: 0.0011
Epoch 63/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9015 - loss: 1.5109 - val_accuracy: 0.8309 -
val_loss: 2.0238 - learning_rate: 0.0011
Epoch 64/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9059 - loss: 1.4620 - val_accuracy: 0.7794 - v
al_loss: 2.0736 - learning_rate: 0.0011
Epoch 65/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9179 - loss: 1.4459 - val_accuracy: 0.7500 -
val_loss: 2.1263 - learning_rate: 0.0011
Epoch 66/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 18s/step - accuracy: 0.8822 - loss: 1.4762
Epoch 66: ReduceLROnPlateau reducing learning rate to 0.0006479999748989939.
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.8884 - loss: 1.4802 - val_accuracy: 0.7684 -
val_loss: 2.1434 - learning_rate: 0.0011
Epoch 67/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9005 - loss: 1.4431 - val_accuracy: 0.7757 -
val_loss: 2.0979 - learning_rate: 6.4800e-04
Epoch 68/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9051 - loss: 1.4419 - val_accuracy: 0.7978 -
val_loss: 2.0343 - learning_rate: 6.4800e-04
Epoch 69/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9146 - loss: 1.4301 - val_accuracy: 0.7904 - v
al_loss: 2.0208 - learning_rate: 6.4800e-04
Epoch 70/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9147 - loss: 1.3968 - val_accuracy: 0.8309 - v
al_loss: 1.9548 - learning_rate: 6.4800e-04
Epoch 71/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9050 - loss: 1.3692 - val_accuracy: 0.8162 - v
al_loss: 1.9161 - learning_rate: 6.4800e-04
Epoch 72/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9295 - loss: 1.3189 - val_accuracy: 0.7904 - v
al_loss: 1.9107 - learning_rate: 6.4800e-04
Epoch 73/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9166 - loss: 1.3120 - val_accuracy: 0.8125 - v
al_loss: 1.8992 - learning_rate: 6.4800e-04
Epoch 74/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9271 - loss: 1.2620 - val_accuracy: 0.8199 -
val_loss: 1.8812 - learning_rate: 6.4800e-04
Epoch 75/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9392 - loss: 1.2535 - val_accuracy: 0.8125 - v
al_loss: 1.8649 - learning_rate: 6.4800e-04
Epoch 76/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9201 - loss: 1.2764 - val_accuracy: 0.8346 - v
al_loss: 1.8255 - learning_rate: 6.4800e-04
Epoch 77/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9233 - loss: 1.2430 - val_accuracy: 0.8235 - v
```

```
al_loss: 1.8236 - learning_rate: 6.4800e-04
Epoch 78/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9240 - loss: 1.2382 - val_accuracy: 0.8199 -
val_loss: 1.8159 - learning_rate: 6.4800e-04
Epoch 79/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.8851 - loss: 1.2805 - val_accuracy: 0.8199 -
val_loss: 1.8800 - learning_rate: 6.4800e-04
Epoch 80/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9285 - loss: 1.2261 - val_accuracy: 0.8235 -
val_loss: 1.8934 - learning_rate: 6.4800e-04
Epoch 81/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 18s/step - accuracy: 0.9330 - loss: 1.1981
Epoch 81: ReduceLROnPlateau reducing learning rate to 0.0003887999919243157.
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9296 - loss: 1.2099 - val_accuracy: 0.7941 -
val_loss: 1.8557 - learning_rate: 6.4800e-04
Epoch 82/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9197 - loss: 1.2184 - val_accuracy: 0.8051 -
val_loss: 1.8428 - learning_rate: 3.8880e-04
Epoch 83/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9204 - loss: 1.2127 - val_accuracy: 0.8272 - v
al_loss: 1.8077 - learning_rate: 3.8880e-04
Epoch 84/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9153 - loss: 1.1873 - val_accuracy: 0.8346 -
val_loss: 1.7701 - learning_rate: 3.8880e-04
Epoch 85/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9058 - loss: 1.2215 - val_accuracy: 0.8125 - v
al_loss: 1.8013 - learning_rate: 3.8880e-04
Epoch 86/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9438 - loss: 1.1165 - val_accuracy: 0.8235 -
val_loss: 1.7428 - learning_rate: 3.8880e-04
Epoch 87/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9182 - loss: 1.1557 - val_accuracy: 0.8309 -
val_loss: 1.7007 - learning_rate: 3.8880e-04
Epoch 88/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9257 - loss: 1.1315 - val_accuracy: 0.8382 - v
al_loss: 1.6543 - learning_rate: 3.8880e-04
Epoch 89/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9417 - loss: 1.1028 - val_accuracy: 0.8419 - v
al_loss: 1.6391 - learning_rate: 3.8880e-04
Epoch 90/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9294 - loss: 1.0887 - val_accuracy: 0.8162 -
val_loss: 1.6563 - learning_rate: 3.8880e-04
Epoch 91/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9232 - loss: 1.0864 - val_accuracy: 0.8309 - v
al_loss: 1.6300 - learning_rate: 3.8880e-04
Epoch 92/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9330 - loss: 1.0674 - val_accuracy: 0.7978 -
val_loss: 1.6828 - learning_rate: 3.8880e-04
Epoch 93/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9358 - loss: 1.0745 - val_accuracy: 0.7794 -
val_loss: 1.6568 - learning_rate: 3.8880e-04
Epoch 94/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9143 - loss: 1.0926 - val_accuracy: 0.8125 -
val_loss: 1.6210 - learning_rate: 3.8880e-04
Epoch 95/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9346 - loss: 1.0245 - val_accuracy: 0.8162 -
val_loss: 1.6111 - learning_rate: 3.8880e-04
Epoch 96/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9342 - loss: 1.0156 - val_accuracy: 0.8309 - v
```

```
al_loss: 1.6154 - learning_rate: 3.8880e-04
Epoch 97/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9112 - loss: 1.0560 - val_accuracy: 0.8309 -
val_loss: 1.5913 - learning_rate: 3.8880e-04
Epoch 98/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9402 - loss: 0.9959 - val_accuracy: 0.8309 - v
al_loss: 1.6003 - learning_rate: 3.8880e-04
Epoch 99/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9270 - loss: 1.0338 - val_accuracy: 0.7978 -
val_loss: 1.6284 - learning_rate: 3.8880e-04
Epoch 100/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9313 - loss: 1.0087 - val_accuracy: 0.8346 - v
al_loss: 1.5760 - learning_rate: 3.8880e-04
Epoch 101/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9163 - loss: 1.0419 - val_accuracy: 0.8015 -
val_loss: 1.6227 - learning_rate: 3.8880e-04
Epoch 102/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9327 - loss: 1.0196 - val_accuracy: 0.8015 - v
al_loss: 1.6051 - learning_rate: 3.8880e-04
Epoch 103/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9383 - loss: 0.9794 - val_accuracy: 0.8051 - v
al_loss: 1.5609 - learning_rate: 3.8880e-04
Epoch 104/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9307 - loss: 0.9776 - val_accuracy: 0.8125 - v
al_loss: 1.5622 - learning_rate: 3.8880e-04
Epoch 105/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9433 - loss: 0.9491 - val_accuracy: 0.7904 - v
al_loss: 1.5776 - learning_rate: 3.8880e-04
Epoch 106/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9420 - loss: 0.9837 - val_accuracy: 0.8199 -
val_loss: 1.5595 - learning_rate: 3.8880e-04
Epoch 107/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9431 - loss: 0.9652 - val_accuracy: 0.8162 -
val_loss: 1.5178 - learning_rate: 3.8880e-04
Epoch 108/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9289 - loss: 0.9673 - val_accuracy: 0.8346 - v
al_loss: 1.4977 - learning_rate: 3.8880e-04
Epoch 109/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9231 - loss: 0.9777 - val_accuracy: 0.8456 - v
al_loss: 1.4976 - learning_rate: 3.8880e-04
Epoch 110/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9231 - loss: 0.9497 - val_accuracy: 0.8603 -
val_loss: 1.5072 - learning_rate: 3.8880e-04
Epoch 111/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9247 - loss: 0.9642 - val_accuracy: 0.8346 - v
al_loss: 1.5339 - learning_rate: 3.8880e-04
Epoch 112/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 453ms/step - accuracy: 0.9200 - loss: 0.9597
Epoch 112: ReduceLROnPlateau reducing learning rate to 0.0002332799951545894.
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9203 - loss: 0.9591 - val_accuracy: 0.8346 - v
al_loss: 1.5335 - learning_rate: 3.8880e-04
Epoch 113/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9352 - loss: 0.9677 - val_accuracy: 0.8162 -
val_loss: 1.5347 - learning_rate: 2.3328e-04
Epoch 114/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9256 - loss: 0.9435 - val_accuracy: 0.8162 - v
al_loss: 1.5445 - learning_rate: 2.3328e-04
Epoch 115/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 18s/step - accuracy: 0.9322 - loss: 0.9936
```

```
Epoch 115: ReduceLROnPlateau reducing learning rate to 0.00013996799534652383.
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9361 - loss: 0.9733 - val_accuracy: 0.8051 -
val_loss: 1.5074 - learning_rate: 2.3328e-04
Epoch 116/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9494 - loss: 0.8873 - val_accuracy: 0.8235 -
val_loss: 1.5068 - learning_rate: 1.3997e-04
Epoch 117/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9368 - loss: 0.9156 - val_accuracy: 0.8419 - v
al_loss: 1.4886 - learning_rate: 1.3997e-04
Epoch 118/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9361 - loss: 0.9054 - val_accuracy: 0.8419 - v
al_loss: 1.4237 - learning_rate: 1.3997e-04
Epoch 119/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9394 - loss: 0.8854 - val_accuracy: 0.8529 -
val_loss: 1.4216 - learning_rate: 1.3997e-04
Epoch 120/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9340 - loss: 0.8987 - val_accuracy: 0.8272 - v
al_loss: 1.4322 - learning_rate: 1.3997e-04
Epoch 121/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9337 - loss: 0.9175 - val_accuracy: 0.8382 -
val_loss: 1.4132 - learning_rate: 1.3997e-04
Epoch 122/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9351 - loss: 0.8788 - val_accuracy: 0.8346 -
val_loss: 1.4046 - learning_rate: 1.3997e-04
Epoch 123/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9537 - loss: 0.8582 - val_accuracy: 0.8493 -
val_loss: 1.3415 - learning_rate: 1.3997e-04
Epoch 124/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9329 - loss: 0.8723 - val_accuracy: 0.8603 - v
al_loss: 1.3562 - learning_rate: 1.3997e-04
Epoch 125/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9531 - loss: 0.8229 - val_accuracy: 0.8346 - v
al_loss: 1.3232 - learning_rate: 1.3997e-04
Epoch 126/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9446 - loss: 0.8369 - val_accuracy: 0.8419 - v
al_loss: 1.3504 - learning_rate: 1.3997e-04
Epoch 127/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9417 - loss: 0.8434 - val_accuracy: 0.8235 -
val_loss: 1.3496 - learning_rate: 1.3997e-04
Epoch 128/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9524 - loss: 0.8106 - val_accuracy: 0.8456 -
val_loss: 1.2980 - learning_rate: 1.3997e-04
Epoch 129/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9530 - loss: 0.8079 - val_accuracy: 0.8346 -
val_loss: 1.3504 - learning_rate: 1.3997e-04
Epoch 130/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9399 - loss: 0.8146 - val_accuracy: 0.8529 - v
al_loss: 1.2545 - learning_rate: 1.3997e-04
Epoch 131/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9509 - loss: 0.8071 - val_accuracy: 0.8162 -
val_loss: 1.3011 - learning_rate: 1.3997e-04
Epoch 132/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9480 - loss: 0.8028 - val_accuracy: 0.8529 -
val_loss: 1.2723 - learning_rate: 1.3997e-04
Epoch 133/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9548 - loss: 0.7921 - val_accuracy: 0.8456 -
val_loss: 1.2533 - learning_rate: 1.3997e-04
Epoch 134/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9651 - loss: 0.7589 - val_accuracy: 0.8603 -
```

```
val_loss: 1.2251 - learning_rate: 1.3997e-04
Epoch 135/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9411 - loss: 0.7790 - val_accuracy: 0.8493 - v
al_loss: 1.2569 - learning_rate: 1.3997e-04
Epoch 136/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9443 - loss: 0.7877 - val_accuracy: 0.8676 - v
al_loss: 1.2439 - learning_rate: 1.3997e-04
Epoch 137/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 18s/step - accuracy: 0.9290 - loss: 0.8007
Epoch 137: ReduceLROnPlateau reducing learning rate to 8.398079371545464e-05.
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9318 - loss: 0.7893 - val_accuracy: 0.8419 -
val_loss: 1.2725 - learning_rate: 1.3997e-04
Epoch 138/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9479 - loss: 0.7513 - val_accuracy: 0.8529 -
val_loss: 1.2239 - learning_rate: 8.3981e-05
Epoch 139/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9642 - loss: 0.7276 - val_accuracy: 0.8272 -
val_loss: 1.2792 - learning_rate: 8.3981e-05
Epoch 140/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9466 - loss: 0.7500 - val_accuracy: 0.8493 -
val_loss: 1.2530 - learning_rate: 8.3981e-05
Epoch 141/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9471 - loss: 0.7514 - val_accuracy: 0.8309 - v
al_loss: 1.2193 - learning_rate: 8.3981e-05
Epoch 142/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9353 - loss: 0.7637 - val_accuracy: 0.8750 -
val_loss: 1.2025 - learning_rate: 8.3981e-05
Epoch 143/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9499 - loss: 0.7450 - val_accuracy: 0.8603 -
val_loss: 1.1896 - learning_rate: 8.3981e-05
Epoch 144/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9562 - loss: 0.7297 - val_accuracy: 0.8787 -
val_loss: 1.1676 - learning_rate: 8.3981e-05
Epoch 145/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9613 - loss: 0.7222 - val_accuracy: 0.8566 - v
al_loss: 1.1777 - learning_rate: 8.3981e-05
Epoch 146/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9490 - loss: 0.7300 - val_accuracy: 0.8676 - v
al_loss: 1.1679 - learning_rate: 8.3981e-05
Epoch 147/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 18s/step - accuracy: 0.9606 - loss: 0.7215
Epoch 147: ReduceLROnPlateau reducing learning rate to 5.038847448304296e-05.
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9550 - loss: 0.7266 - val_accuracy: 0.8529 -
val_loss: 1.1773 - learning_rate: 8.3981e-05
Epoch 148/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9515 - loss: 0.7173 - val_accuracy: 0.8603 - v
al_loss: 1.1535 - learning_rate: 5.0388e-05
Epoch 149/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9386 - loss: 0.7539 - val_accuracy: 0.8566 -
val_loss: 1.1304 - learning_rate: 5.0388e-05
Epoch 150/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9506 - loss: 0.7195 - val_accuracy: 0.8529 - v
al_loss: 1.1423 - learning_rate: 5.0388e-05
Epoch 151/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9543 - loss: 0.7184 - val_accuracy: 0.8419 -
val_loss: 1.1463 - learning_rate: 5.0388e-05
Epoch 152/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 503ms/step - accuracy: 0.9511 - loss: 0.7062
Epoch 152: ReduceLROnPlateau reducing learning rate to 3.023308381671086e-05.
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9512 - loss: 0.7060 - val_accuracy: 0.8493 - v
al_loss: 1.1751 - learning_rate: 5.0388e-05
Epoch 153/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9575 - loss: 0.6958 - val_accuracy: 0.8566 - v
al_loss: 1.1424 - learning_rate: 3.0233e-05
Epoch 154/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9530 - loss: 0.6919 - val_accuracy: 0.8713 -
val_loss: 1.1315 - learning_rate: 3.0233e-05
Epoch 155/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 446ms/step - accuracy: 0.9550 - loss: 0.6927
Epoch 155: ReduceLROnPlateau reducing learning rate to 1.8139850726583973e-05.
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9547 - loss: 0.6933 - val_accuracy: 0.8640 - v
al_loss: 1.1350 - learning_rate: 3.0233e-05
Epoch 156/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9573 - loss: 0.6883 - val_accuracy: 0.8603 -
val_loss: 1.1231 - learning_rate: 1.8140e-05
Epoch 157/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9542 - loss: 0.7121 - val_accuracy: 0.8603 -
val_loss: 1.1140 - learning_rate: 1.8140e-05
Epoch 158/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9503 - loss: 0.7009 - val_accuracy: 0.8603 - v
al_loss: 1.1110 - learning_rate: 1.8140e-05
Epoch 159/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9506 - loss: 0.6924 - val_accuracy: 0.8640 -
val_loss: 1.1327 - learning_rate: 1.8140e-05
Epoch 160/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9519 - loss: 0.6943 - val_accuracy: 0.8456 - v
al_loss: 1.1225 - learning_rate: 1.8140e-05
Epoch 161/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 469ms/step - accuracy: 0.9675 - loss: 0.6696
Epoch 161: ReduceLROnPlateau reducing learning rate to 1.0883909999392926e-05.
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9667 - loss: 0.6713 - val_accuracy: 0.8750 - v
al_loss: 1.1171 - learning_rate: 1.8140e-05
Epoch 162/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9524 - loss: 0.7030 - val_accuracy: 0.8676 -
val_loss: 1.0639 - learning_rate: 1.0884e-05
Epoch 163/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9548 - loss: 0.6808 - val_accuracy: 0.8897 -
val_loss: 1.0624 - learning_rate: 1.0884e-05
Epoch 164/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9626 - loss: 0.6757 - val_accuracy: 0.8529 - v
al_loss: 1.0831 - learning_rate: 1.0884e-05
Epoch 165/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9521 - loss: 0.6801 - val_accuracy: 0.8676 - v
al_loss: 1.0630 - learning_rate: 1.0884e-05
Epoch 166/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 496ms/step - accuracy: 0.9483 - loss: 0.6891
Epoch 166: ReduceLROnPlateau reducing learning rate to 6.530346217914484e-06.
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9484 - loss: 0.6887 - val_accuracy: 0.8382 - v
al_loss: 1.0936 - learning_rate: 1.0884e-05
Epoch 167/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9616 - loss: 0.6776 - val_accuracy: 0.8934 -
val_loss: 1.0396 - learning_rate: 6.5303e-06
Epoch 168/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9466 - loss: 0.6625 - val_accuracy: 0.8566 -
val_loss: 1.0593 - learning_rate: 6.5303e-06
Epoch 169/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9502 - loss: 0.6997 - val_accuracy: 0.8529 - v
al_loss: 1.0942 - learning_rate: 6.5303e-06
```

```
Epoch 170/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 482ms/step - accuracy: 0.9569 - loss: 0.6836
Epoch 170: ReduceLROnPlateau reducing learning rate to 5e-06.
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9566 - loss: 0.6849 - val_accuracy: 0.8640 - v
al_loss: 1.0522 - learning_rate: 6.5303e-06
Epoch 171/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9506 - loss: 0.6849 - val_accuracy: 0.8640 - v
al_loss: 1.0686 - learning_rate: 5.0000e-06
Epoch 172/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9499 - loss: 0.6826 - val_accuracy: 0.8824 - v
al_loss: 1.0339 - learning_rate: 5.0000e-06
Epoch 173/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9575 - loss: 0.6730 - val_accuracy: 0.8713 - v
al_loss: 1.0452 - learning_rate: 5.0000e-06
Epoch 174/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9373 - loss: 0.7034 - val_accuracy: 0.8860 -
val_loss: 1.0191 - learning_rate: 5.0000e-06
Epoch 175/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9367 - loss: 0.7085 - val_accuracy: 0.8566 -
val_loss: 1.0121 - learning_rate: 5.0000e-06
Epoch 176/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9575 - loss: 0.6584 - val_accuracy: 0.8456 - v
al_loss: 1.0279 - learning_rate: 5.0000e-06
Epoch 177/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9574 - loss: 0.6814 - val_accuracy: 0.8456 -
val_loss: 1.0706 - learning_rate: 5.0000e-06
Epoch 178/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9537 - loss: 0.6896 - val_accuracy: 0.8750 - v
al_loss: 1.0160 - learning_rate: 5.0000e-06
Epoch 179/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9726 - loss: 0.6472 - val_accuracy: 0.8824 - v
al_loss: 1.0165 - learning_rate: 5.0000e-06
Epoch 180/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9575 - loss: 0.6782 - val_accuracy: 0.8529 - v
al_loss: 1.0389 - learning_rate: 5.0000e-06
Epoch 181/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9686 - loss: 0.6702 - val_accuracy: 0.8787 -
val_loss: 1.0265 - learning_rate: 5.0000e-06
Epoch 182/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9607 - loss: 0.6640 - val_accuracy: 0.8529 - v
al_loss: 1.0124 - learning_rate: 5.0000e-06
Epoch 183/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9597 - loss: 0.6635 - val_accuracy: 0.8787 -
val_loss: 0.9966 - learning_rate: 5.0000e-06
Epoch 184/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9572 - loss: 0.6759 - val_accuracy: 0.8566 -
val_loss: 1.0364 - learning_rate: 5.0000e-06
Epoch 185/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 23s/step - accuracy: 0.9555 - loss: 0.6634 - val_accuracy: 0.8603 -
val_loss: 1.0328 - learning_rate: 5.0000e-06
Epoch 186/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9598 - loss: 0.6727 - val_accuracy: 0.8713 - v
al_loss: 1.0002 - learning_rate: 5.0000e-06
Epoch 187/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9541 - loss: 0.6744 - val_accuracy: 0.8934 -
val_loss: 0.9663 - learning_rate: 5.0000e-06
Epoch 188/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9607 - loss: 0.6726 - val_accuracy: 0.8860 - v
al_loss: 1.0040 - learning_rate: 5.0000e-06
```

```
Epoch 189/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9594 - loss: 0.6750 - val_accuracy: 0.8603 - v
al_loss: 0.9961 - learning_rate: 5.0000e-06
Epoch 190/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9572 - loss: 0.6653 - val_accuracy: 0.8676 - v
al_loss: 1.0286 - learning_rate: 5.0000e-06
Epoch 191/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9455 - loss: 0.6845 - val_accuracy: 0.8640 - v
al_loss: 0.9971 - learning_rate: 5.0000e-06
Epoch 192/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9484 - loss: 0.6740 - val_accuracy: 0.8456 - v
al_loss: 1.0242 - learning_rate: 5.0000e-06
Epoch 193/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9588 - loss: 0.6612 - val_accuracy: 0.8676 - v
al_loss: 1.0537 - learning_rate: 5.0000e-06
Epoch 194/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9559 - loss: 0.6656 - val_accuracy: 0.8750 - v
al_loss: 1.0075 - learning_rate: 5.0000e-06
Epoch 195/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9600 - loss: 0.6673 - val_accuracy: 0.8603 - v
al_loss: 1.0402 - learning_rate: 5.0000e-06
Epoch 196/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 25s 24s/step - accuracy: 0.9618 - loss: 0.6567 - val_accuracy: 0.8603 -
val_loss: 1.0217 - learning_rate: 5.0000e-06
Epoch 197/250
2/2 ━━━━━━━━━━━━━━━━━━━━ 24s 6s/step - accuracy: 0.9475 - loss: 0.6704 - val_accuracy: 0.8640 - v
al_loss: 1.0225 - learning_rate: 5.0000e-06
```

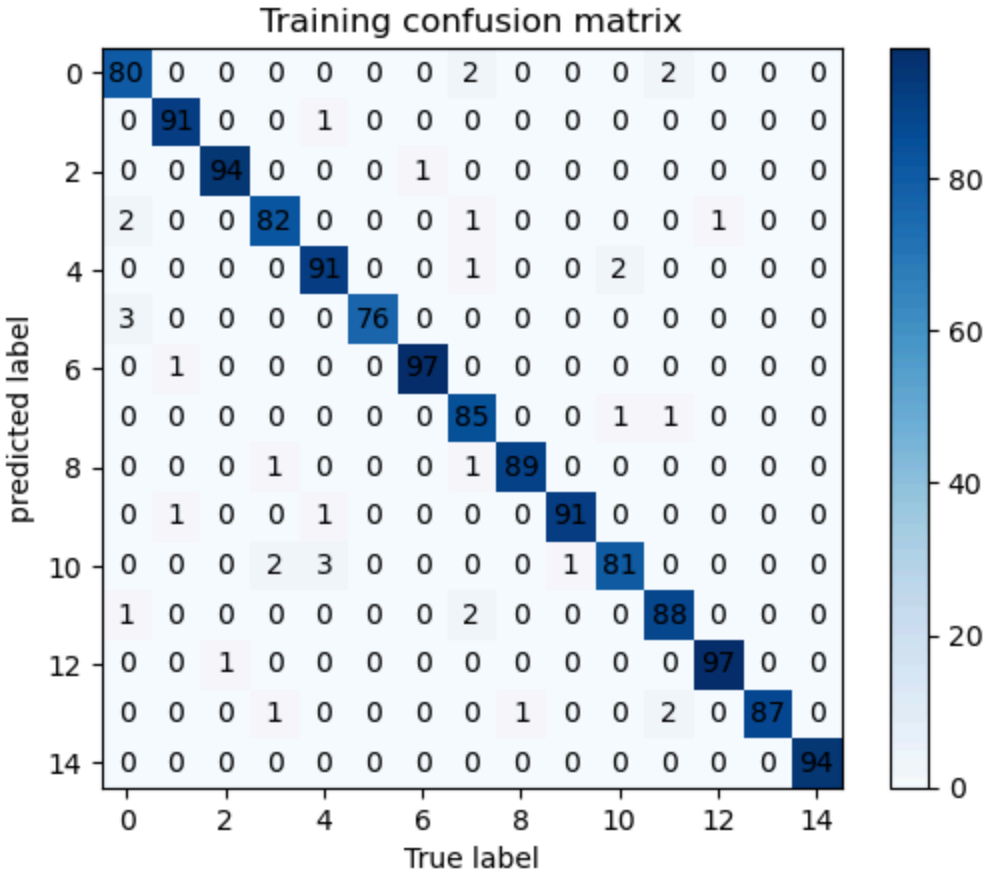
In [24]:

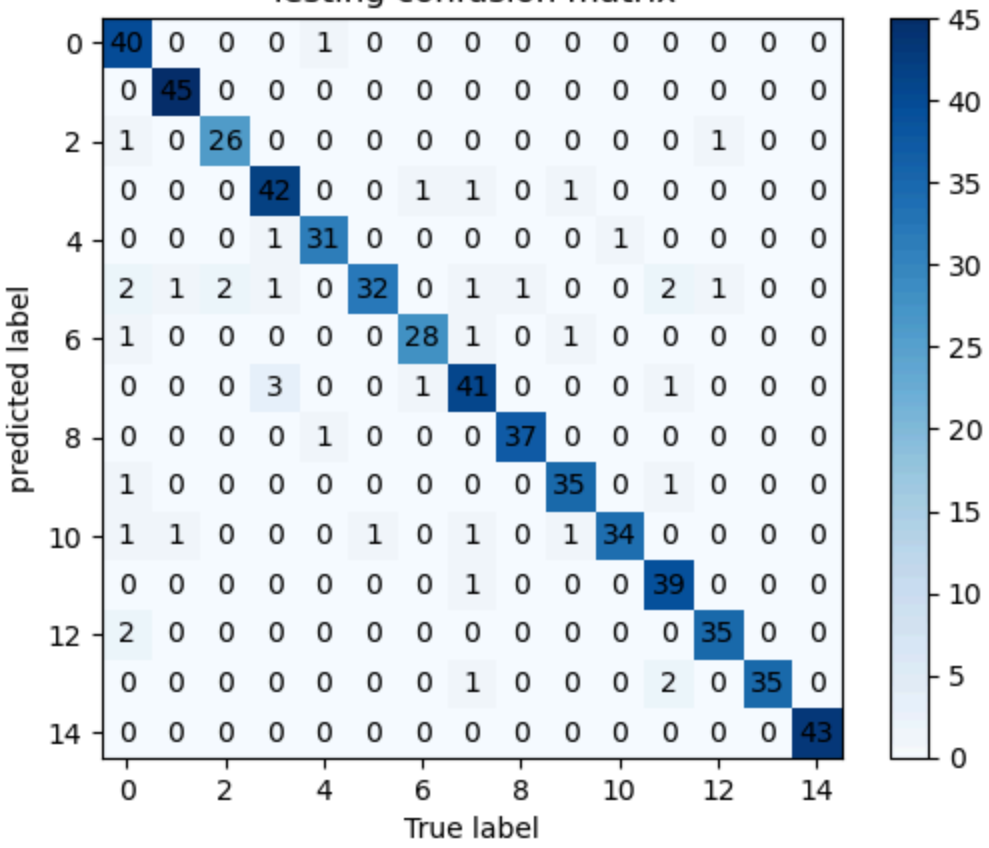
```python
#I'm just going to make a function that produces all the outputs and plots, should have done this

def diagnostics(model, hist, epochs ):
    train_acc = hist.history['accuracy']
    val_acc = hist.history['val_accuracy']

    epochs_ind = [i for i in range(1, 1 + len(train_acc))]

    plt.plot(epochs_ind, train_acc, 'blue', label = 'Training accuracy')
    plt.plot(epochs_ind, val_acc, 'red', label = 'Validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.title('Accuracy over training epochs')
    plt.legend()
    plt.show()

    _, train_acc = model.evaluate(X_train, Y_train, verbose = 0)
    _, test_acc = model.evaluate(X_test, Y_test, verbose = 0)

    print(f"Training accuracy: {train_acc*100}%")
    print(f"Testing accuracy: {test_acc*100}%")

    #confusion matrix

    train_preds = np.argmax(model.predict(X_train), axis = 1)
    test_preds = np.argmax(model.predict(X_test), axis = 1)

    train_cm = confusion_matrix(np.argmax(Y_train, axis = 1), train_preds)
    test_cm = confusion_matrix(np.argmax(Y_test, axis = 1), test_preds)

    plt.imshow(train_cm, cmap = plt.cm.Blues)
    plt.colorbar()
```

```
    for i in range(train_cm.shape[0]):
      for j in range(train_cm.shape[1]):
        plt.text(j, i, train_cm[i, j], ha = 'center', va = 'center')
    plt.title('Training confusion matrix')
    plt.xlabel('True label')
    plt.ylabel('predicted label')
    plt.show()

    plt.imshow(test_cm, cmap = plt.cm.Blues)
    plt.colorbar()
    for i in range(test_cm.shape[0]):
      for j in range(test_cm.shape[1]):
        plt.text(j, i, test_cm[i, j], ha = 'center', va = 'center')
    plt.title('Testing confusion matrix')
    plt.xlabel('True label')
    plt.ylabel('predicted label')
    plt.show()
```

In [25]:
```
diagnostics(animal_tx_3, history_tx_3, epochs)
```


```
Training accuracy: 97.27941155433655%
Testing accuracy: 92.97945499420166%
43/43 ━━━━━━━━━━━━━━━━━━━━ 11s 198ms/step
19/19 ━━━━━━━━━━━━━━━━━━━━ 3s 155ms/step
```





I am overall happy with this model, but I would like to try and replicate this behavior without the use of transfer learning, so I will attempt one final CNN with more depth, and more regularization to account for overfitting.

In [41]:
```
from keras.layers import SpatialDropout2D
from keras.layers import SeparableConv2D
```

Here I had to change rom Conv2D to separableConv2D CNN layers because the model could not run

In [50]:
```
train_gen_3 = generator2.flow(X_train, Y_train, subset = "training", batch_size = 500)
val_gen_3 = generator2.flow(X_train, Y_train, subset = "validation", batch_size = 500)
```

In [53]:
```
final_cnn = Sequential()

final_cnn.add(InputLayer(input_shape = X_shape[1:]))

def cnn_block(model, f, k, drop, l1, l2, act):
    model.add(SeparableConv2D(f,
                      kernel_size = k,
                      padding = 'same',
                      activation = None,
                      depthwise_regularizer = regularizers.l1_l2(l1, l2)))
    model.add(BatchNormalization())
    model.add(Activation(act))
    #using a different dropout layer that should work better for CNN
    model.add(SpatialDropout2D(drop))
    model.add(MaxPooling2D(pool_size = (2, 2)))

filters = [128, 128, 64, 32, 32]

for f in filters:
    cnn_block(final_cnn, f, 3, 0.5, 0.001, 0.001, 'relu')


final_cnn.add(GlobalAveragePooling2D())
final_cnn.add(Dense(128,
                        activation = 'relu',
                        kernel_regularizer = regularizers.l2(0.0005)))
final_cnn.add(Dropout(0.3))

final_cnn.add(Dense(64,
                        activation = 'relu',
                        kernel_regularizer = regularizers.l2(0.0005)))
final_cnn.add(Dropout(0.3))

final_cnn.add(Dense(32,
                        activation = 'relu',
                        kernel_regularizer = regularizers.l2(0.0005)))
final_cnn.add(Dropout(0.3))

final_cnn.add(Dense(Y_shape[1], activation = 'softmax'))

optimizer = Adam(learning_rate = 0.002, amsgrad = True)

final_cnn.compile(loss = 'categorical_crossentropy',
                      optimizer = optimizer,
                      metrics = ['accuracy'])

final_cnn.summary()
```

Model: "sequential_6"

Layer (type)	Output Shape	Param #
separable_conv2d_16 (SeparableConv2D)	(None, 224, 224, 128)	539
batch_normalization_25 (BatchNormalization)	(None, 224, 224, 128)	512
activation_25 (Activation)	(None, 224, 224, 128)	0
spatial_dropout2d_25 (SpatialDropout2D)	(None, 224, 224, 128)	0
max_pooling2d_25 (MaxPooling2D)	(None, 112, 112, 128)	0
separable_conv2d_17 (SeparableConv2D)	(None, 112, 112, 128)	17,664
batch_normalization_26 (BatchNormalization)	(None, 112, 112, 128)	512
activation_26 (Activation)	(None, 112, 112, 128)	0
spatial_dropout2d_26 (SpatialDropout2D)	(None, 112, 112, 128)	0
max_pooling2d_26 (MaxPooling2D)	(None, 56, 56, 128)	0
separable_conv2d_18 (SeparableConv2D)	(None, 56, 56, 64)	9,408
batch_normalization_27 (BatchNormalization)	(None, 56, 56, 64)	256
activation_27 (Activation)	(None, 56, 56, 64)	0
spatial_dropout2d_27 (SpatialDropout2D)	(None, 56, 56, 64)	0
max_pooling2d_27 (MaxPooling2D)	(None, 28, 28, 64)	0
separable_conv2d_19 (SeparableConv2D)	(None, 28, 28, 32)	2,656
batch_normalization_28 (BatchNormalization)	(None, 28, 28, 32)	128
activation_28 (Activation)	(None, 28, 28, 32)	0
spatial_dropout2d_28 (SpatialDropout2D)	(None, 28, 28, 32)	0
max_pooling2d_28 (MaxPooling2D)	(None, 14, 14, 32)	0
separable_conv2d_20 (SeparableConv2D)	(None, 14, 14, 32)	1,344
batch_normalization_29	(None, 14, 14, 32)	128

(BatchNormalization)		
activation_29 (Activation)	(None, 14, 14, 32)	0
spatial_dropout2d_29 (SpatialDropout2D)	(None, 14, 14, 32)	0
max_pooling2d_29 (MaxPooling2D)	(None, 7, 7, 32)	0
global_average_pooling2d_5 (GlobalAveragePooling2D)	(None, 32)	0
dense_20 (Dense)	(None, 128)	4,224
dropout_15 (Dropout)	(None, 128)	0
dense_21 (Dense)	(None, 64)	8,256
dropout_16 (Dropout)	(None, 64)	0
dense_22 (Dense)	(None, 32)	2,080
dropout_17 (Dropout)	(None, 32)	0
dense_23 (Dense)	(None, 15)	495

Total params: 48,202 (188.29 KB)

Trainable params: 47,434 (185.29 KB)

Non-trainable params: 768 (3.00 KB)

In [ ]:
```
#reduces learning rate if loss value plateaus for 3 iterations
reduce_2 = ReduceLROnPlateau(
    monitor = 'val_loss',
    patience = 3,
    verbose = 1,
    factor = 0.6,
    min_lr = 0.000005
)

#early stopping to prevent some overfitting
early_stop = EarlyStopping(monitor = 'val_loss',
                           patience = 10, #higher than the LR reduction
                           restore_best_weights = True)

#train the model
epochs = 200 #More epochs since early stopping should prevent reaching if overfitting starts
batch_size = 32
#using same epochs and batches and callbacks as above. Also using same generalization as above
history_final = final_cnn.fit(train_gen_3,
                              epochs = epochs,
                              batch_size = batch_size,
                             validation_data = val_gen_3,
                            callbacks = [reduce_2, early_stop])
```

```
Epoch 1/200
```

In [ ]:
```
diagnostics(final_cnn, history_final, epochs)
```

The increased depth simply will not work. The final model then is `animal_tx_3`, a transfer learning model built on top of MobileNetV2. This model uses an additional Dense layer, and starts with a very high learning rate which decays over time. This model additionally employs early stopping, which would stop the model should it plateau for an extended period. This model takes as input generated images that employ randomization techniques on the original dataset, including rotations, shifts, flipping, and zoom. A new batch of generated images was then used every epoch. The inputs are scaled to all have value between 0 and 1. This improve generalization greatly from when I first implemented it, and adding increased randomization seemed to help the final chosen model generalize even over many epochs, and actually adding epochs seemed to help with generalization as the random image generator would continue to add noise so the model was forced to train to the differences. I believe that if I had more computing power available to me I could have created a better model without transfer learning, but to add the depth I feel is necessary to increase accuracy properly, the training becomes far too resource intensive and caused my kernel to crash repeatedly, even when using more lightweight layers. The level at which my machine could complete the necessary computations was simply insufficient in either producing extensive overfitting, or presenting very low overall accuracy even over many epochs. While the final model still is somewhat lacking in accuracy, having a split in accuracy over testing and training data around 4.3% is impressive alongside obtaining over 90% accuracy on both when comparing to initial models. I believe that given more computing power, we could unfreeze some of the layers in MobileNetV2 or add additional layers ourselves outside of the transfer learning model to create an even better model at predicting classes.

In the diagnostic plots you can see oscillations in the learning curves. In both curves this is to be expected, as the input is different every epoch. With the validation curve this is especially expected as the dataset changes and trains to the set that is generated for that epoch. You can also see over time the learning curves slowly converge together, even as the training curve appears to level off. This is due to the excess epochs allowing the model to become further generalized using the random image generator for the inputs.

The confusion matrices are also substantially better in this model than preceeding models, showing very low rates of misclassification.

We also see higher accuracy on both the original training data as well as the testing data than you see in the learning curves. This is again because we are using a random image generator in the input for fitting the model, and so the model does an excellent job of classifying the images when there are no transformations

A likely reason that this was so difficult is that there were many classes of different animals, some of which resemble each other in many ways. Additionally, there were, in my opinion, not enough images of each type to help the model differentiate between similar classes. Using the image generator did significantly help, but was not enough without implementing MobileNetV2. If I were to do this again with a more powerful machine, I would implement gaussian noise layers to aid in generalization, as well as more depth. I would again use the random image generator, as adding this was certainly a turning point in the overall training. I beleive that given a dataset, if you are able to fit a distribution to it in much the same way the random image generator does, and instead sample from the distribution rather than directly from the dataset each epoch, you are able to obtain a much more generalized model than you would otherwise.

This project showcases the need for heavy generalization in conjunction with deep models to achieve high accuracy when a complex dataset is being modeled. This project also shows some of the limitations neural networks have, in particular neural networks, especially in image classification models, require very high powered machines to complete the fitting process. I began this project in Google colab, but then had to

change gears to use the high end computing desktop offered by the university in order to fit even the more simple models in any reasonable amount of time. While Convolutional Neural Networks excel at classifying images, much better than most classical machine learning approaches, they are limited by the computing power at hand.

In [30]:
```
!{sys.executable} -m pip install --upgrade pandoc
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandoc in c:\users\lcleymaet\appdata\roaming\python\python312\site
-packages (2.4)
Requirement already satisfied: plumbum in c:\users\lcleymaet\appdata\roaming\python\python312\sit
e-packages (from pandoc) (1.9.0)
Requirement already satisfied: ply in c:\programdata\anaconda3\lib\site-packages (from pandoc)
(3.11)
Requirement already satisfied: pywin32 in c:\programdata\anaconda3\lib\site-packages (from plumbu
m->pandoc) (308)
```

In [ ]:
```

```