3c

March 28, 2021

0.1 CSc 84020 Neural Networks and Deep Learning, Spring 2021

Homework 2 (3c)

Andrea Ceres and Shao Liu

0.2 Import Libraries & Load Data

Import libraries and define classes

[ ]:

```
from google.colab import drive
import glob
import os
import numpy as np
import time
import random as rn
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from itertools import cycle
import pylab
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc, roc_auc_score,␣
 ↪precision_recall_curve
from sklearn.metrics import classification_report as cr
from sklearn.metrics import confusion_matrix as cm
import tensorflow as tf
from tensorflow.keras.callbacks import Callback
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, BatchNormalization, Dropout,␣
 ↪LeakyReLU
from tensorflow.keras.regularizers import L2
from tensorflow.keras.initializers import GlorotNormal
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.utils import to_categorical, plot_model

print(tf.__version__)
```

2.4.1

[ ]:
```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
start_total_time = time.time()
```

[ ]:
```
# Different options for classes: selection to be set with constants

# critters (6 classes)
critters = ['ant', 'bee', 'butterfly', 'mosquito', 'scorpion',
           'spider']

# birds (6 classes)
birds = ['bird', 'duck', 'flamingo', 'owl', 'parrot',
           'penguin']

# ocean animals (8 classes)
ocean_animals = ['crab', 'dolphin', 'fish', 'lobster', 'octopus',
           'sea%20turtle', 'shark', 'whale']

# land mammals (22 classes)
land_mammals = ['bear', 'camel', 'cat', 'cow', 'dog',
           'elephant', 'giraffe', 'hedgehog', 'horse', 'kangaroo',
           'lion', 'monkey', 'mouse', 'panda', 'pig',
           'rabbit', 'raccoon', 'rhinoceros', 'sheep', 'squirrel',
           'tiger', 'zebra']

# all animals (47 classes)
all_animals = ['ant', 'bat', 'bear', 'bee', 'bird',
           'butterfly', 'camel', 'cat', 'cow', 'crab',
           'crocodile', 'dog', 'dolphin', 'duck', 'elephant',
           'fish', 'flamingo', 'frog', 'giraffe', 'hedgehog',
           'horse', 'kangaroo', 'lion', 'lobster', 'monkey',
           'mosquito', 'mouse', 'octopus', 'owl', 'panda',
           'parrot', 'penguin', 'pig', 'rabbit', 'raccoon',
           'rhinoceros', 'scorpion', 'sea%20turtle', 'shark', 'sheep',
           'snail', 'snake', 'spider', 'squirrel',  'tiger',
           'whale', 'zebra']
```

Set constants

[ ]:
```
classes = critters
CLASS_SAMPLE_MAX = 25000
DATA_DIR = '/content/drive/My Drive/Colab Notebooks/NNDL/HW2/data/'
```

Load data

[ ]:

```python
def load_bitmaps(data_dir=DATA_DIR, class_sample_max=CLASS_SAMPLE_MAX):
    class_files = []
    for c in classes:
        print(c, end=' ')
        class_files.append(os.path.join(DATA_DIR, c + '.npy'))
    class_files.sort()

    X = np.empty([0,784])
    y = np.empty([0])

    print()
    for id, class_file in enumerate(class_files):
        print(id, end=' ')
        loaded_data = np.load(class_file)
        loaded_data = loaded_data[0:CLASS_SAMPLE_MAX, :]
        labels = np.full(loaded_data.shape[0],id)

        X = np.concatenate((X, loaded_data), axis = 0)
        y = np.append(y, labels)

    return X, y
```

[ ]:

```python
start = time.time()
X, y = load_bitmaps()
print(f'\nLoading time: %.3f' % int(time.time() - start), 'seconds')
print(y[-CLASS_SAMPLE_MAX-5:-CLASS_SAMPLE_MAX+5])
```

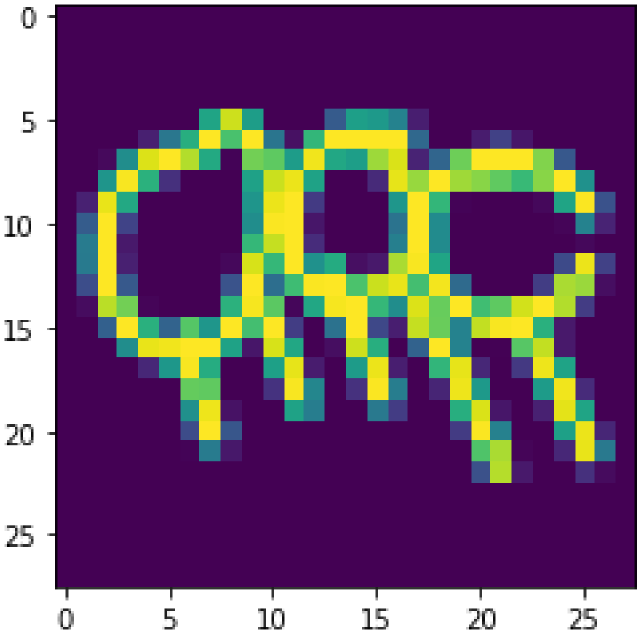
```
ant bee butterfly mosquito scorpion spider
0 1 2 3 4 5
Loading time: 12.000 seconds
[4. 4. 4. 4. 4. 5. 5. 5. 5. 5.]
```

Data example

[ ]:

```python
id = 20000
plt.imshow(X[id].reshape(28,28))
print(classes[int(y[id].item())])
```

```
ant
```



```
[ ]: X[20000]
```

```
[ ]: array([  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0., 149., 236., 142.,   0.,   0.,   0.,  73.,
           144., 137., 113.,  21.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,  22., 101., 160., 255.,
           182., 255.,  99.,  11., 169., 255., 255., 255., 255.,  86.,   0.,
             0.,  19.,  49.,  15.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             6., 125., 242., 255., 226., 155.,   2., 201., 192., 141., 250.,
           153., 143., 218., 241.,  25.,  83., 199., 255., 255., 255., 207.,
            70.,   0.,   0.,   0.,   0.,   0., 133., 253., 163.,  35.,   0.,
             0.,   0., 146., 235., 248., 147.,   0.,   0.,  30., 246., 216.,
           255., 219., 210., 193., 172., 210., 255., 129.,   0.,   0.,   0.,
            23., 247., 154.,   0.,   0.,   0.,   0.,   0., 133., 248., 255.,
            43.,   0.,   0.,   0., 134., 255., 169.,   2.,   0.,   0.,   0.,
             5., 149., 254.,  64.,   0.,   0.,  74., 255.,  51.,   0.,   0.,
             0.,   0.,   0., 125., 254., 255.,  15.,   0.,   0.,   0.,  97.,
           255., 123.,   0.,   0.,   0.,   0.,   0.,   3., 113.,  25.,   0.,
             0., 106., 255.,  17.,   0.,   0.,   0.,   0.,   0., 170., 233.,
           255.,  32.,   0.,   0.,   0., 110., 255., 123.,   0.,   0.,   0.,
             0.,   0.,   0.,   5.,   0.,   0.,   0., 104., 255.,  22.,   0.,
             0.,   0.,   0.,   5., 240., 170., 255., 129., 157.,  10.,  17.,
           223., 253., 150.,   0.,   0.,   0.,   0.,   0.,  57., 248.,  49.,
             0.,   0.,  60., 255.,  71.,   0.,   0.,   0.,   0.,  65., 255.,
            92., 176., 254., 255., 183., 235., 247., 179., 251.,  88.,   0.,
             0.,   0.,  46., 223., 217.,   9.,   0.,   0.,   8., 229., 201.,
             3.,   0.,   0.,   0., 164., 252., 191.,   4., 153., 252., 255.,
           170., 125., 242., 196., 255., 177., 192., 238., 255., 226.,  45.,
             0.,   0.,   0.,   0.,  78., 255., 162.,  81., 188., 134., 255.,
           179., 255.,  46.,   0.,  96., 255.,  55.,  21., 233., 197., 113.,
           232., 253., 255., 162.,  17.,   0.,   0.,   0.,   0.,   0.,   0.,
           125., 247., 250., 255., 255., 147.,  15., 240., 153.,   0.,   9.,
           236., 160.,   0.,  80., 255., 113.,   0.,   1., 189., 232.,  17.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,  28., 135., 253., 158.,
             0.,   0., 145., 246.,  19.,   0., 140., 248.,  21.,   0., 168.,
           246.,  28.,   0.,  42., 249., 148.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0., 195., 193.,   0.,   0.,  37., 253., 117.,
             0.,  36., 254., 108.,   0.,  26., 249., 137.,   0.,   0., 140.,
           250.,  31.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 128.,
           249.,  12.,   0.,   0., 147., 108.,   0.,   0., 103.,  45.,   0.,
             0., 160., 241.,  15.,   0.,  23., 245., 149.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,  59., 255.,  68.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,  44., 254., 112.,   0.,
             0., 145., 248.,  27.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             2., 106.,  16.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0., 185., 223.,   3.,   0.,  28., 247., 103.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  63., 227.,
            17.,   0.,   0.,  35.,   7.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
             0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
            0.,    0.,    0.])
```

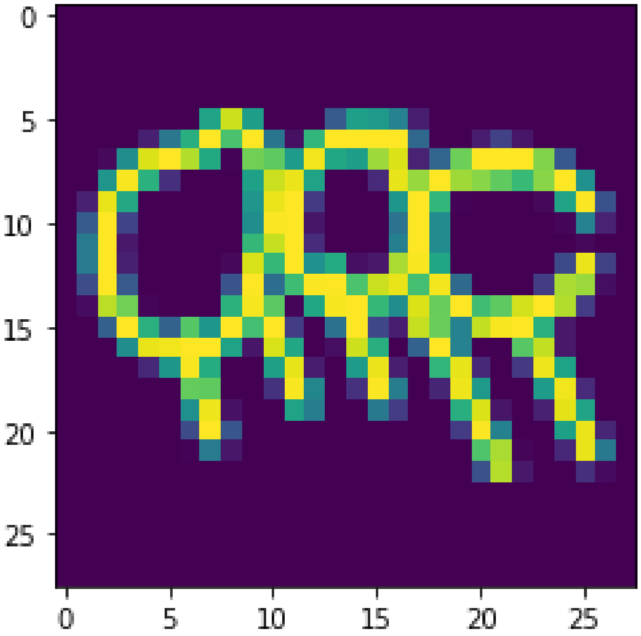
0.3 Normalization

```
[ ]: # Normalization to range [0, 1]
     X = X / 255
```

Data example after normalization

```
[ ]: id = 20000
     plt.imshow(X[id].reshape(28,28))
     print(classes[int(y[id].item())])
```

```
ant
```



```
[ ]: X[20000]
```

```
[ ]: array([0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.58431373, 0.9254902 , 0.55686275,
            0.        , 0.        , 0.        , 0.28627451, 0.56470588,
            0.5372549 , 0.44313725, 0.08235294, 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.08627451, 0.39607843, 0.62745098,
            1.        , 0.71372549, 1.        , 0.38823529, 0.04313725,
            0.6627451 , 1.        , 1.        , 1.        , 1.        ,
            0.3372549 , 0.        , 0.        , 0.0745098 , 0.19215686,
            0.05882353, 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.02352941, 0.49019608,
            0.94901961, 1.        , 0.88627451, 0.60784314, 0.00784314,
            0.78823529, 0.75294118, 0.55294118, 0.98039216, 0.6       ,
            0.56078431, 0.85490196, 0.94509804, 0.09803922, 0.3254902 ,
            0.78039216, 1.        , 1.        , 1.        , 0.81176471,
            0.2745098 , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.52156863, 0.99215686, 0.63921569, 0.1372549 ,
            0.        , 0.        , 0.        , 0.57254902, 0.92156863,
            0.97254902, 0.57647059, 0.        , 0.        , 0.11764706,
            0.96470588, 0.84705882, 1.        , 0.85882353, 0.82352941,
            0.75686275, 0.6745098 , 0.82352941, 1.        , 0.50588235,
            0.        , 0.        , 0.        , 0.09019608, 0.96862745,
            0.60392157, 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.52156863, 0.97254902, 1.        , 0.16862745,
            0.        , 0.        , 0.        , 0.5254902 , 1.        ,
            0.6627451 , 0.00784314, 0.        , 0.        , 0.        ,
            0.01960784, 0.58431373, 0.99607843, 0.25098039, 0.        ,
            0.        , 0.29019608, 1.        , 0.2       , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.49019608,
            0.99607843, 1.        , 0.05882353, 0.        , 0.        ,
            0.        , 0.38039216, 1.        , 0.48235294, 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.01176471,
            0.44313725, 0.09803922, 0.        , 0.        , 0.41568627,
            1.        , 0.06666667, 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.66666667, 0.91372549, 1.        ,
            0.1254902 , 0.        , 0.        , 0.        , 0.43137255,
            1.        , 0.48235294, 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.01960784, 0.        ,
            0.        , 0.        , 0.40784314, 1.        , 0.08627451,
            0.        , 0.        , 0.        , 0.        , 0.01960784,
            0.94117647, 0.66666667, 1.        , 0.50588235, 0.61568627,
            0.03921569, 0.06666667, 0.8745098 , 0.99215686, 0.58823529,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.22352941, 0.97254902, 0.19215686, 0.        , 0.        ,
            0.23529412, 1.        , 0.27843137, 0.        , 0.        ,
            0.        , 0.        , 0.25490196, 1.        , 0.36078431,
            0.69019608, 0.99607843, 1.        , 0.71764706, 0.92156863,
            0.96862745, 0.70196078, 0.98431373, 0.34509804, 0.        ,
            0.        , 0.        , 0.18039216, 0.8745098 , 0.85098039,
            0.03529412, 0.        , 0.        , 0.03137255, 0.89803922,
            0.78823529, 0.01176471, 0.        , 0.        , 0.        ,
            0.64313725, 0.98823529, 0.74901961, 0.01568627, 0.6       ,
            0.98823529, 1.        , 0.66666667, 0.49019608, 0.94901961,
            0.76862745, 1.        , 0.69411765, 0.75294118, 0.93333333,
            1.        , 0.88627451, 0.17647059, 0.        , 0.        ,
            0.        , 0.        , 0.30588235, 1.        , 0.63529412,
            0.31764706, 0.7372549 , 0.5254902 , 1.        , 0.70196078,
            1.        , 0.18039216, 0.        , 0.37647059, 1.        ,
            0.21568627, 0.08235294, 0.91372549, 0.77254902, 0.44313725,
            0.90980392, 0.99215686, 1.        , 0.63529412, 0.06666667,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.49019608, 0.96862745, 0.98039216, 1.        ,
            1.        , 0.57647059, 0.05882353, 0.94117647, 0.6       ,
            0.        , 0.03529412, 0.9254902 , 0.62745098, 0.        ,
            0.31372549, 1.        , 0.44313725, 0.        , 0.00392157,
            0.74117647, 0.90980392, 0.06666667, 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.10980392, 0.52941176, 0.99215686, 0.61960784, 0.        ,
            0.        , 0.56862745, 0.96470588, 0.0745098 , 0.        ,
            0.54901961, 0.97254902, 0.08235294, 0.        , 0.65882353,
            0.96470588, 0.10980392, 0.        , 0.16470588, 0.97647059,
            0.58039216, 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.76470588, 0.75686275, 0.        , 0.        , 0.14509804,
            0.99215686, 0.45882353, 0.        , 0.14117647, 0.99607843,
            0.42352941, 0.        , 0.10196078, 0.97647059, 0.5372549 ,
            0.        , 0.        , 0.54901961, 0.98039216, 0.12156863,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.50196078, 0.97647059,
            0.04705882, 0.        , 0.        , 0.57647059, 0.42352941,
            0.        , 0.        , 0.40392157, 0.17647059, 0.        ,
            0.        , 0.62745098, 0.94509804, 0.05882353, 0.        ,
            0.09019608, 0.96078431, 0.58431373, 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.23137255, 1.        , 0.26666667, 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.17254902,
            0.99607843, 0.43921569, 0.        , 0.        , 0.56862745,
            0.97254902, 0.10588235, 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.00784314,
            0.41568627, 0.0627451 , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.7254902 , 0.8745098 ,
            0.01176471, 0.        , 0.10980392, 0.96862745, 0.40392157,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.24705882, 0.89019608, 0.06666667, 0.        ,
            0.        , 0.1372549 , 0.02745098, 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        ])
```

0.4 Multi-layer Perceptron Model

Set constants and parameters

[ ]:
```python
RANDOM_STATE = 84020

NUM_CLASSES = len(classes)
INPUT_DIM = X.shape[1]

TEST_SIZE = 0.1
VALIDATION_SPLIT = 0.22

VERBOSE = 0         # choose from: [0, 1]
BATCH_SIZE = 128    # choose from: [32, 64, 128]
EPOCHS = 30         # choose from: [30, 40]
```

[ ]:
```python
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
optimizer = Adam(lr=0.001, epsilon=1e-06)
                      # choose from: [sgd, Adam(lr=0.001, epsilon=1e-06)]

activation = LeakyReLU(alpha=0.01) # choose from: 'relu', LeakyReLU(alpha=0.01)

# dropout = 0         # choose from: [0, 0.1, 0.2, 0.4]

# l2_val = 0          # choose from: [0, 0.1, 0.0005]

loss = 'sparse_categorical_crossentropy'

metrics = ['sparse_categorical_accuracy']
```

Split X and y into train and test sets

[ ]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                  test_size=TEST_SIZE,
                                                  shuffle=True,
                                                  random_state=RANDOM_STATE)
```

[ ]:

```
print('Number of classes:', NUM_CLASSES)
print('Input dimension:', INPUT_DIM)
print('Total dataset \t\tX:\t', X.shape, '\t\ty:\t', y.shape)
print(100 * (1 - TEST_SIZE), '% train \t\tX_train:', X_train.shape,␣
 ↪'\t\ty_train:',y_train.shape)
print(100 * TEST_SIZE, '% test \t\tX_test:\t', X_test.shape, '\t\ty_test:
 ↪\t',y_test.shape)
```

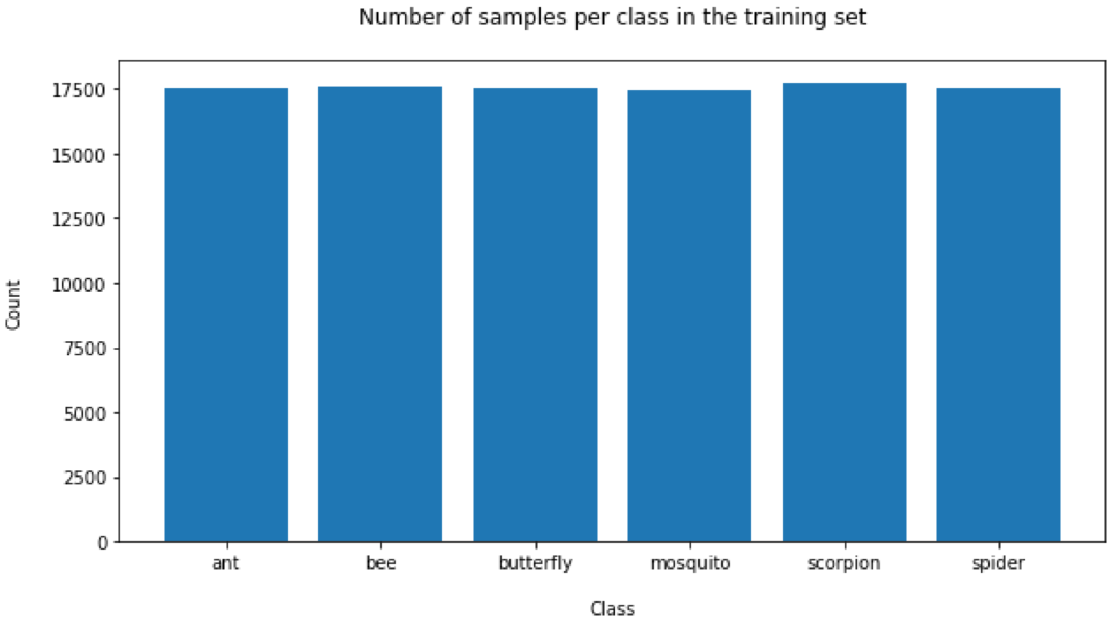
```
Number of classes: 6
Input dimension: 784
Total dataset            X:       (150000, 784)          y:       (150000,)
90.0 % train             X_train: (135000, 784)          y_train: (135000,)
10.0 % test              X_test:  (15000, 784)           y_test:  (15000,)
```

[ ]:

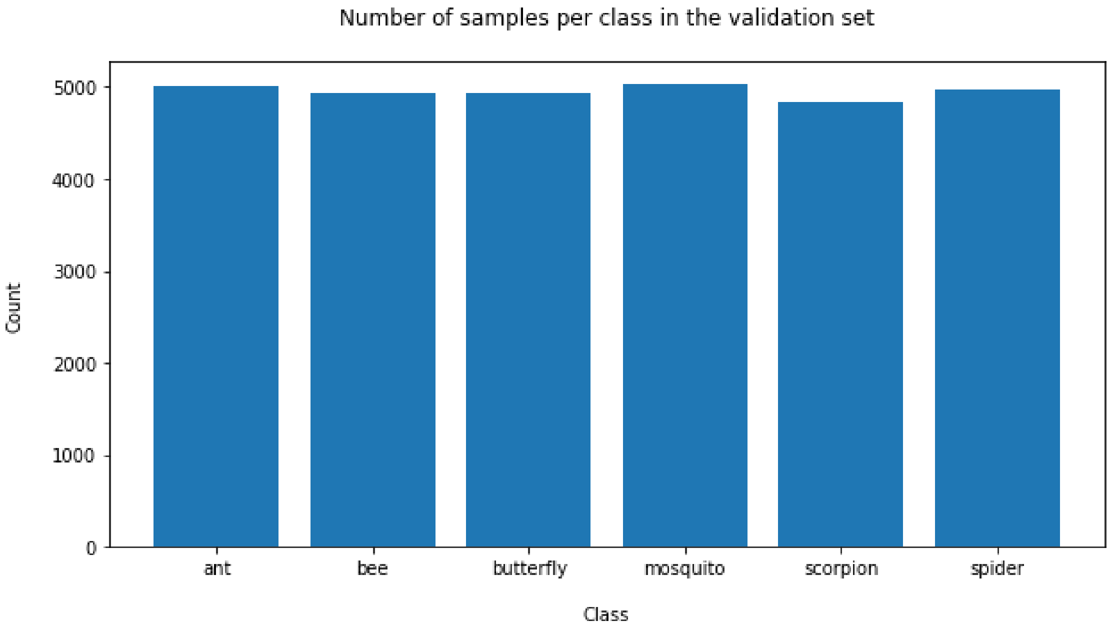
```
# Classification: split the dataset into train, validate and test
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=TEST_SIZE,␣
 ↪random_state=RANDOM_STATE)
Xtrain, Xval, Ytrain, Yval = train_test_split(Xtrain, Ytrain,␣
 ↪test_size=VALIDATION_SPLIT, random_state=RANDOM_STATE)
```

[ ]:

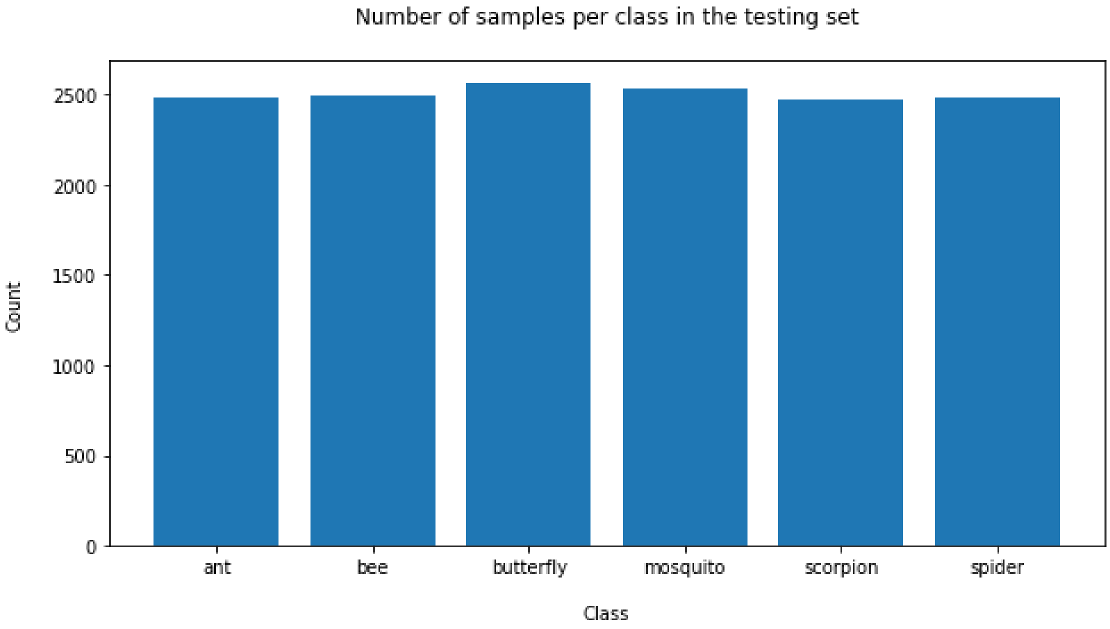
```
# Training data
unique, counts = np.unique(Ytrain, return_counts=True)
pylab.rcParams['figure.figsize'] = (10,5)
plt.bar(critters, counts)
plt.title('Number of samples per class in the training set\n')
plt.ylabel('Count\n')
plt.xlabel('\nClass')
plt.show()
```



[ ]:
```
# Validation data
unique, counts = np.unique(Yval, return_counts=True)
pylab.rcParams['figure.figsize'] = (10,5)
plt.bar(critters, counts)
plt.title('Number of samples per class in the validation set\n')
plt.ylabel('Count\n')
plt.xlabel('\nClass')
plt.show()
```



[ ]:
```
# Testing data
unique, counts = np.unique(Ytest, return_counts=True)
pylab.rcParams['figure.figsize'] = (10,5)
plt.bar(critters, counts)
plt.title('Number of samples per class in the testing set\n')
plt.ylabel('Count\n')
plt.xlabel('\nClass')
plt.show()
```



Model functions

```
[ ]: def build_model(dropout, l2_val, activation=activation):

         if type(activation) != str:
             activ = 'LeakyReLU'
         else:
             activ = activation

         model = Sequential(name='Multi_Layer_Perceptron_{}_{}DO_{}L2'.format(activ,␣
      ↪dropout, l2_val))
         model.add(Input(shape=(INPUT_DIM,),
                         name='Input'))
         model.add(Dense(32, activation=activation,
                         kernel_initializer='glorot_normal',
                         bias_initializer='zeros',
                         kernel_regularizer=L2(l2_val),
                         name='Dense1_{}'.format(activ)))
         model.add(BatchNormalization(name='BatchNormalization1'))
         model.add(Dense(32, activation=activation,
                         kernel_initializer='glorot_normal',
                         bias_initializer='zeros',
                         kernel_regularizer=L2(l2_val),
                         name='Dense2_{}'.format(activ)))
         model.add(Dropout(dropout, name='Dropout_{}'.format(dropout)))
         model.add(Dense(32, activation=activation,
                         kernel_initializer='glorot_normal',
                         bias_initializer='zeros',
                         kernel_regularizer=L2(l2_val),
                         name='Dense3_{}'.format(activ)))
         model.add(Dense(32, activation=activation,
                         kernel_initializer='glorot_normal',
                         bias_initializer='zeros',
                         kernel_regularizer=L2(l2_val),
                         name='Dense4_{}'.format(activ)))
         model.add(BatchNormalization(name='BatchNormalization2'))
         model.add(Dense(16, activation=activation,
                         kernel_initializer='glorot_normal',
                         bias_initializer='zeros',
                         kernel_regularizer=L2(l2_val),
                         name='Dense5_{}'.format(activ)))
         model.add(Dense(NUM_CLASSES, activation='softmax',
                         name='Output'))

         return model
```

[ ]:

```
def compile_model(model, loss=loss, optimizer=optimizer, metrics=metrics):
    model.compile(loss=loss, optimizer=optimizer, metrics=metrics)
```

[ ]:

```
def fit_model(model):
    start_time = time.time()
    history = model.fit(X_train, y_train,
                        epochs=EPOCHS,
                        batch_size=BATCH_SIZE,
                        validation_split=VALIDATION_SPLIT,
                        verbose=VERBOSE)
    time_taken = time.time() - start_time
    print(f'Total train time for {EPOCHS} epochs = %.3f' % time_taken,␣
 ↪'seconds\n\n')

    return history
```

[ ]:

```
def evaluate_model(model):
    score = model.evaluate(X_test, y_test,
                           batch_size=BATCH_SIZE,
                           verbose=VERBOSE)

    return score
```

[ ]:

```
def predict_model(model):
    y_pred = model.predict(X_test, batch_size=BATCH_SIZE)

    return y_pred
```

[ ]:

```python
def plot_accuracy(history, score, metrics=metrics):
    plt.plot(history.history[metrics[0]])
    plt.plot(history.history['val_'+metrics[0]])

    plt.title('Training and validation accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.yticks(ticks=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
    plt.legend(['Training accuracy', 'Validation accuracy'], loc='best')
    plt.show()

    print(f'Test accuracy: {score[1]:.4}\n\n')
```

[ ]:

```python
def plot_loss(history, score):
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])

    plt.title('Training and validation loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.yticks(ticks=[0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2])
    plt.legend(['Training loss', 'Validation loss'], loc='best')
    plt.show()

    print(f'Test loss: {score[0]:.4}\n\n')
```

[ ]:

```python
def print_classification_report(y_pred):
    print(cr(y_test, y_pred.argmax(axis=1), target_names=classes))
```

[ ]:

```python
def plot_confusion_matrix(y_pred):
    # source: https://stackoverflow.com/questions/20927368/
 ↪how-to-normalize-a-confusion-matrix
    cmatrix = cm(y_test, y_pred.argmax(axis=1))
    cmatrix_norm = cmatrix.astype('float') / cmatrix.sum(axis=1)[:, np.newaxis]
    fig, ax = plt.subplots(figsize=(8,4))
    sns.heatmap(cmatrix_norm, annot=True, fmt="0.3f", xticklabels=classes,␣
 ↪yticklabels=classes)
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title('Normalized confusion matrix')
    plt.show(block=False)
```

[ ]:

```python
def plot_precision_recall_curve(y_pred):
    n_classes = len(classes)
    y_test_ohe = to_categorical(y_test, num_classes=NUM_CLASSES)
    # source: https://stackoverflow.com/questions/56090541/
↪how-to-plot-precision-and-recall-of-multiclass-classifier
    # precision recall curve
    precision = dict()
    recall = dict()
    for i in range(n_classes):
        precision[i], recall[i], _ = precision_recall_curve(y_test_ohe[:, i],
                                                            y_pred[:, i])
        plt.plot(recall[i], precision[i], lw=1, label='{}'.format(classes[i]))

    plt.xlabel("Recall")
    plt.ylabel("Precision")
    plt.legend(loc="best")
    plt.title("Precision-Recall curve")
    plt.show()
```

[ ]:

```
def plot_roc_curve(y_pred):
    # source: https://stackoverflow.com/questions/64924911/
↪plotting-multiclass-roc-curve
    # Compute ROC curve and ROC area for each class
    n_classes = len(classes)
    y_test_ohe = to_categorical(y_test, num_classes=NUM_CLASSES)
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test_ohe[:, i], y_pred[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    # Compute micro-average ROC curve and ROC area
    fpr["micro"], tpr["micro"], _ = roc_curve(y_test_ohe.ravel(), y_pred.
↪ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    lw=1

    # First aggregate all false positive rates
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

    # Then interpolate all ROC curves at this points
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
    mean_tpr /= n_classes
    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure()
    plt.plot(fpr["micro"], tpr["micro"],
            label='micro-average ROC curve (area = {0:0.2f})'
                ''.format(roc_auc["micro"]),
            color='deeppink', linestyle=':', linewidth=4)

    plt.plot(fpr["macro"], tpr["macro"],
            label='macro-average ROC curve (area = {0:0.2f})'
                ''.format(roc_auc["macro"]),
            color='cornflowerblue', linestyle=':', linewidth=4)

    colors = cycle(['darkgreen', 'darkorange', 'navy', 'hotpink', 'maroon',␣
↪'cyan'])
    for i, color in zip(range(n_classes), colors):
        plt.plot(fpr[i], tpr[i], color=color, lw=lw,
                label='ROC curve of class {0} (area = {1:0.2f})'
                ''.format(classes[i], roc_auc[i]))

    plt.plot([0, 1], [0, 1], 'k--', lw=lw)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Some extension of Receiver operating characteristic to␣
↪multi-class')
    plt.legend(loc="lower right")
    plt.show()
```

[ ]:

```
def print_auc_scores(y_pred):
    # source:https://scikit-learn.org/stable/auto_examples/model_selection/
↪plot_roc.html
    macro_roc_auc_ovo = roc_auc_score(y_test, y_pred, multi_class="ovo",
                                    average="macro")
    weighted_roc_auc_ovo = roc_auc_score(y_test, y_pred, multi_class="ovo",
                                        average="weighted")
    macro_roc_auc_ovr = roc_auc_score(y_test, y_pred, multi_class="ovr",
                                    average="macro")
    weighted_roc_auc_ovr = roc_auc_score(y_test, y_pred, multi_class="ovr",
                                        average="weighted")
    print("One-vs-One ROC AUC scores:\n{:.6f} (macro),\n{:.6f} "
        "(weighted by prevalence)"
        .format(macro_roc_auc_ovo, weighted_roc_auc_ovo))
    print("One-vs-Rest ROC AUC scores:\n{:.6f} (macro),\n{:.6f} "
        "(weighted by prevalence)"
        .format(macro_roc_auc_ovr, weighted_roc_auc_ovr))
```

[ ]:
```
def plot_model_png(model, dropout, l2_val, optimizer):
    if optimizer == sgd:
        plot_path = '../MLP_sgd_'+str(dropout)+'DO_'+str(l2_val)+'L2.png'
    elif type(optimizer) != str:
        plot_path = '../MLP_adam_'+str(dropout)+'DO_'+str(l2_val)+'L2.png'
    else:
        plot_path = '../MLP_'+optimizer+'_'+str(dropout)+'DO_'+str(l2_val)+'L2.
↪png'
    plot_model(model, to_file=os.path.join(DATA_DIR, plot_path),␣
↪show_shapes=True)
```

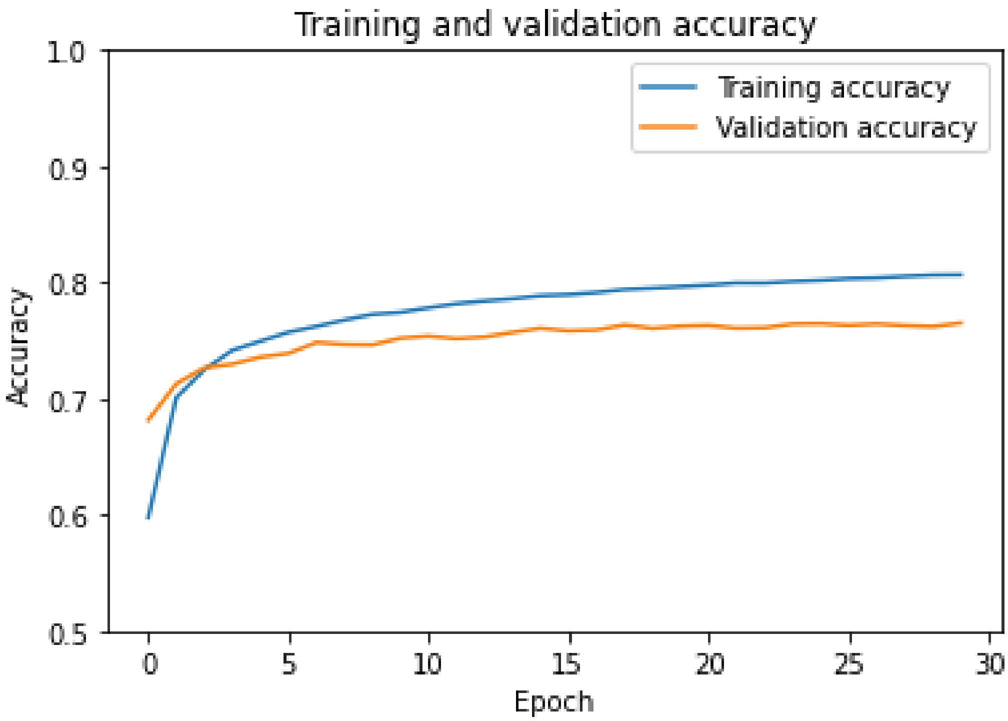
[ ]:
```
def run_model(dropout=0, l2_val=0, activation=activation,
              loss=loss, optimizer=optimizer, metrics=metrics):
    model = build_model(dropout, l2_val, activation)
    model.summary()
    plot_model_png(model, dropout, l2_val, optimizer)
    compile_model(model)
    history = fit_model(model)
    score = evaluate_model(model)
    y_pred = predict_model(model)
    plot_accuracy(history, score)
    plot_loss(history, score)
    print_classification_report(y_pred)
    print('\n\n')
    plot_confusion_matrix(y_pred)
    print('\n\n')
    plot_precision_recall_curve(y_pred)
    print('\n\n')
    plot_roc_curve(y_pred)
    print('\n\n')
    print_auc_scores(y_pred)
    print('\n\n')

    return model, history, score, y_pred
```

0.5 Run models

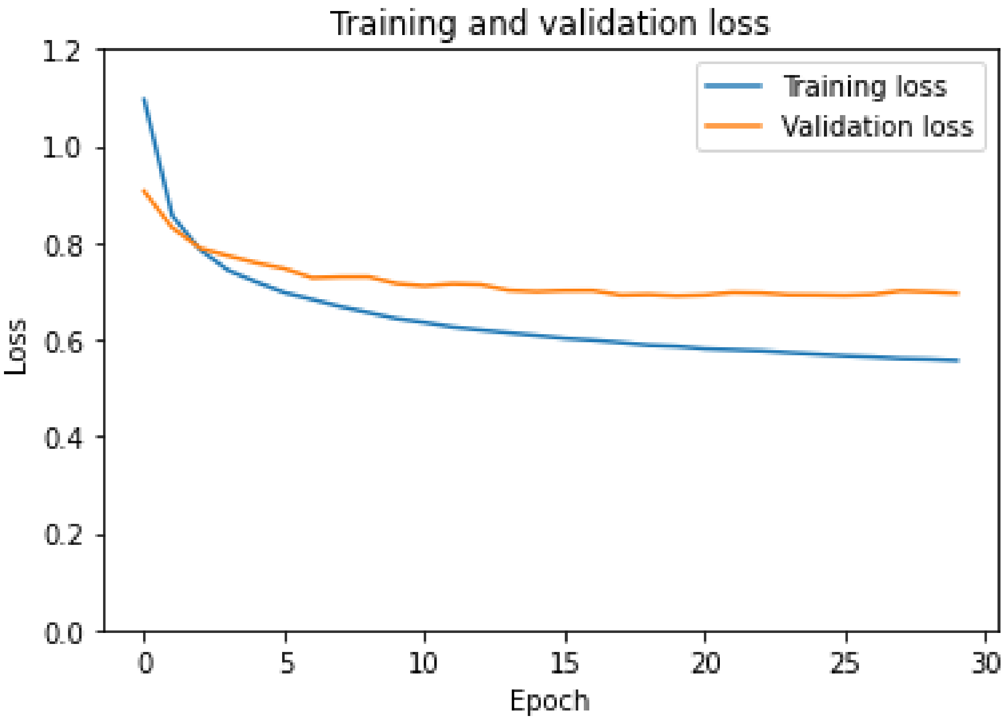
[ ]:
```
model_base, history_base, score_base, y_pred_base = run_model()
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0D0_0L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120

_________________________________________________________________
BatchNormalization1 (BatchNo (None, 32)                128

_________________________________________________________________
Dense2_LeakyReLU (Dense)     (None, 32)                1056

_________________________________________________________________
Dropout_0 (Dropout)          (None, 32)                0

_________________________________________________________________
Dense3_LeakyReLU (Dense)     (None, 32)                1056

_________________________________________________________________
Dense4_LeakyReLU (Dense)     (None, 32)                1056

_________________________________________________________________
BatchNormalization2 (BatchNo (None, 32)                128

_________________________________________________________________
Dense5_LeakyReLU (Dense)     (None, 16)                528

_________________________________________________________________
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128

_________________________________________________________________
Total train time for 30 epochs = 71.950 seconds
```

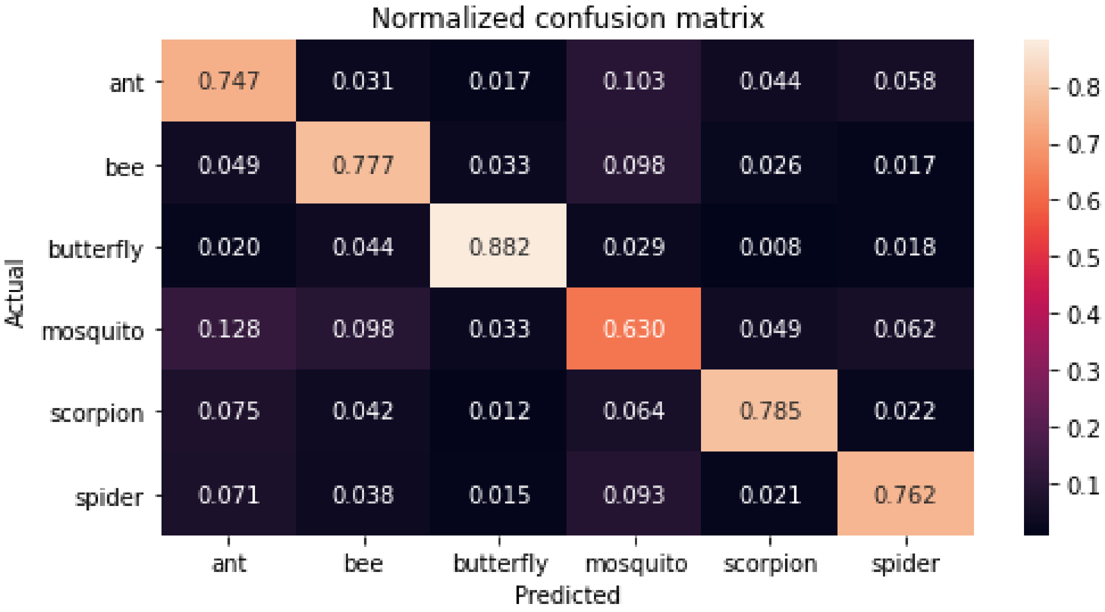


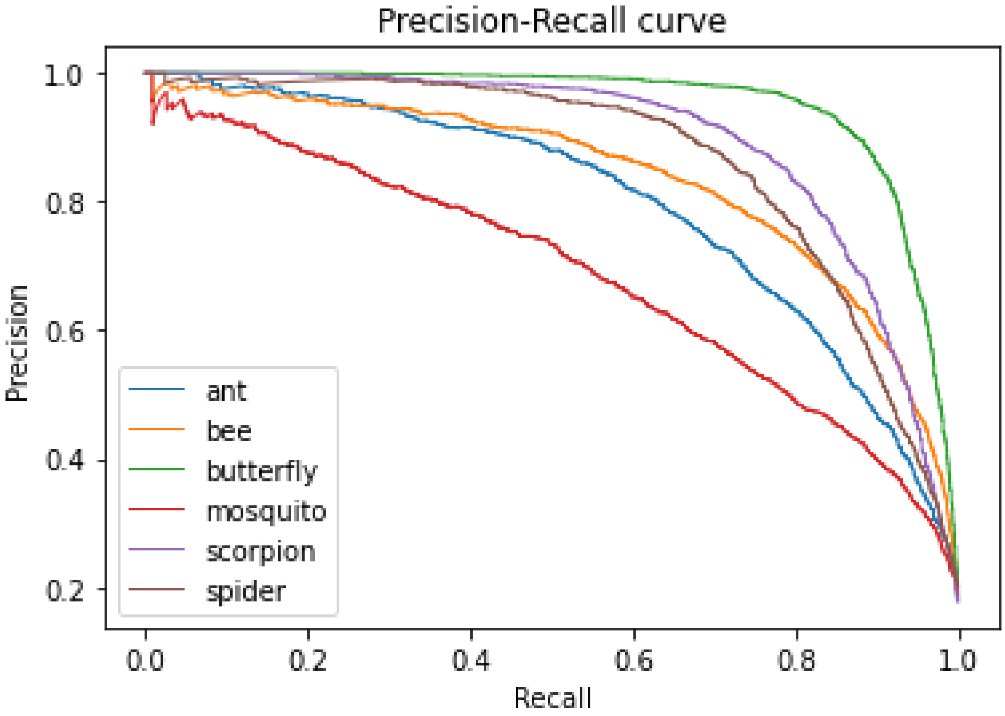
```
Test accuracy: 0.7641
```

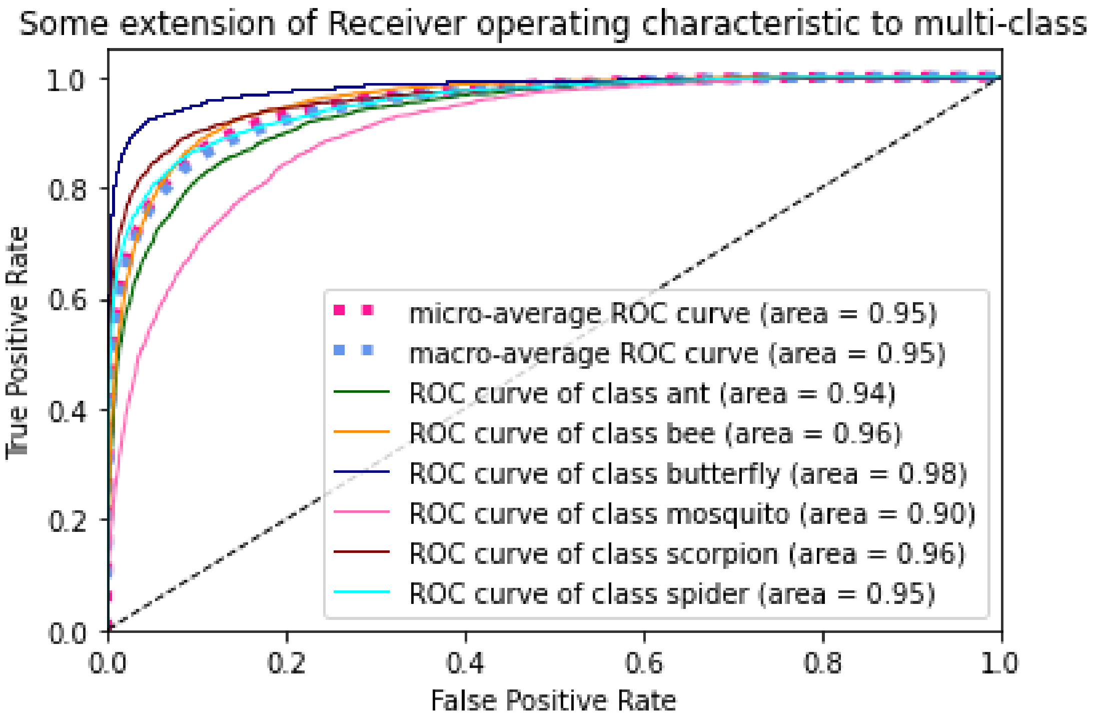


```
Test loss: 0.6984
```

```
              precision    recall  f1-score   support

         ant       0.68      0.75      0.71      2478
         bee       0.75      0.78      0.76      2488
   butterfly       0.89      0.88      0.89      2557
    mosquito       0.62      0.63      0.63      2527
    scorpion       0.84      0.79      0.81      2468
      spider       0.81      0.76      0.79      2482

    accuracy                           0.76     15000
   macro avg       0.77      0.76      0.76     15000
weighted avg       0.77      0.76      0.77     15000
```


Normalized confusion matrix
Actual
Predicted
ant 0.747 0.031 0.017 0.103 0.044 0.058
bee 0.049 0.777 0.033 0.098 0.026 0.017
butterfly 0.020 0.044 0.882 0.029 0.008 0.018
mosquito 0.128 0.098 0.033 0.630 0.049 0.062
scorpion 0.075 0.042 0.012 0.064 0.785 0.022
spider 0.071 0.038 0.015 0.093 0.021 0.762
ant bee butterfly mosquito scorpion spider
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Precision-Recall curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
ant
bee
butterfly
mosquito
scorpion
spider



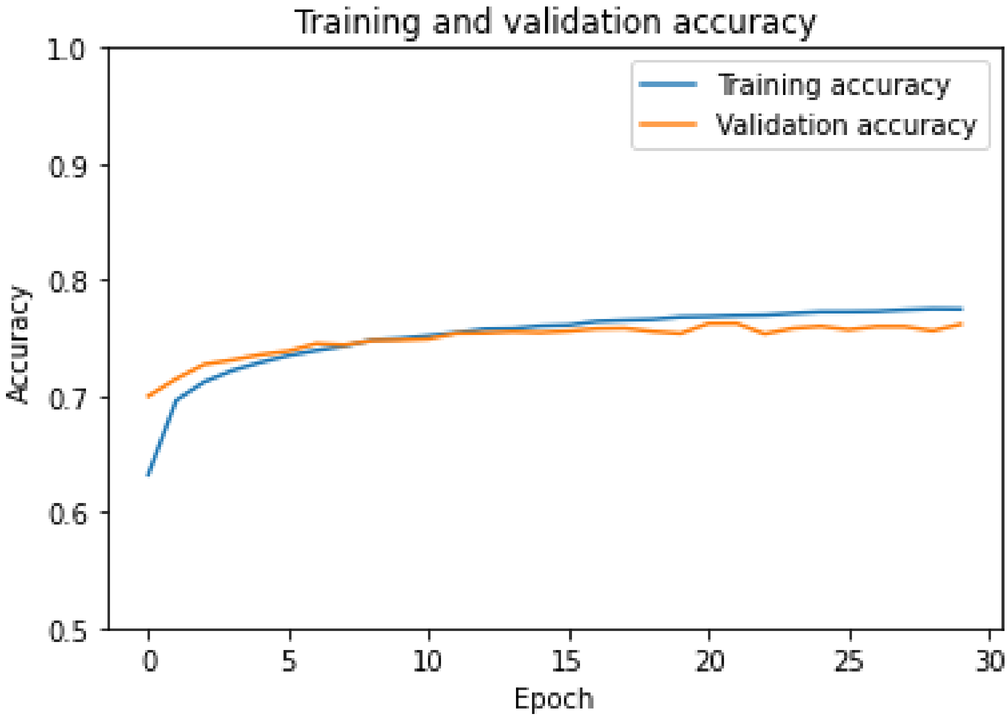
```
One-vs-One ROC AUC scores:
0.948369 (macro),
0.948412 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.948432 (macro),
0.948461 (weighted by prevalence)
```

[ ]:

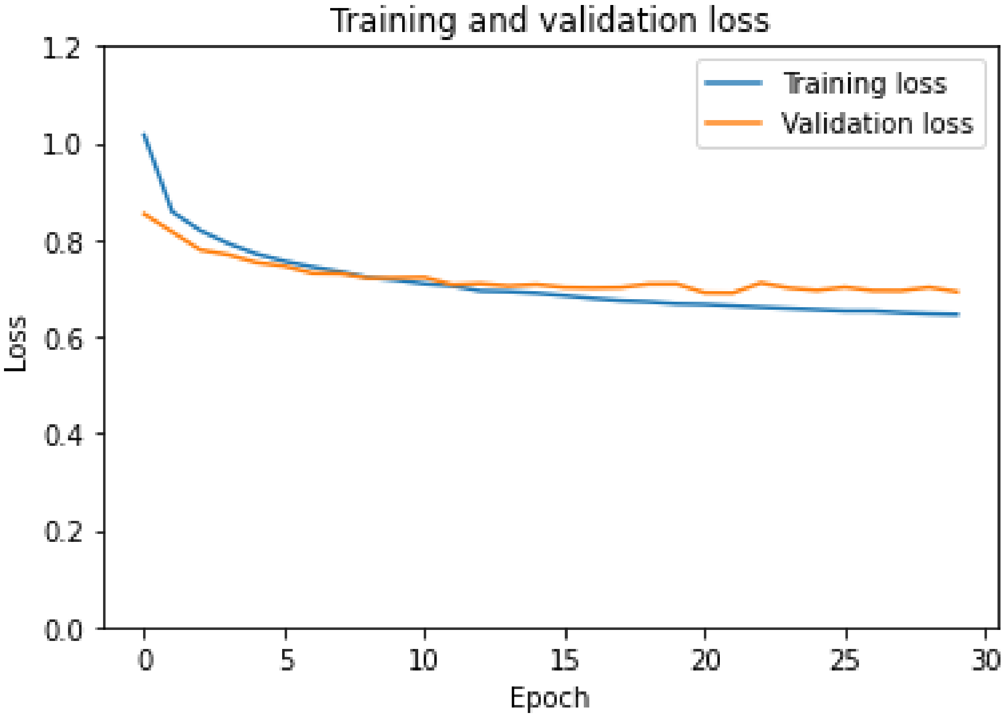
```
model_10do, history_10do, score_10do, y_pred_10do = run_model(dropout=0.1)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.1DO_0L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120     

_________________________________________________________________
BatchNormalization1 (BatchNo (None, 32)                128       

_________________________________________________________________
Dense2_LeakyReLU (Dense)     (None, 32)                1056      

_________________________________________________________________
Dropout_0.1 (Dropout)        (None, 32)                0         

_________________________________________________________________
Dense3_LeakyReLU (Dense)     (None, 32)                1056      

_________________________________________________________________
Dense4_LeakyReLU (Dense)     (None, 32)                1056      

_________________________________________________________________
BatchNormalization2 (BatchNo (None, 32)                128       

_________________________________________________________________
Dense5_LeakyReLU (Dense)     (None, 16)                528       

_________________________________________________________________
Output (Dense)               (None, 6)                 102       
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128

_________________________________________________________________
Total train time for 30 epochs = 70.748 seconds
```

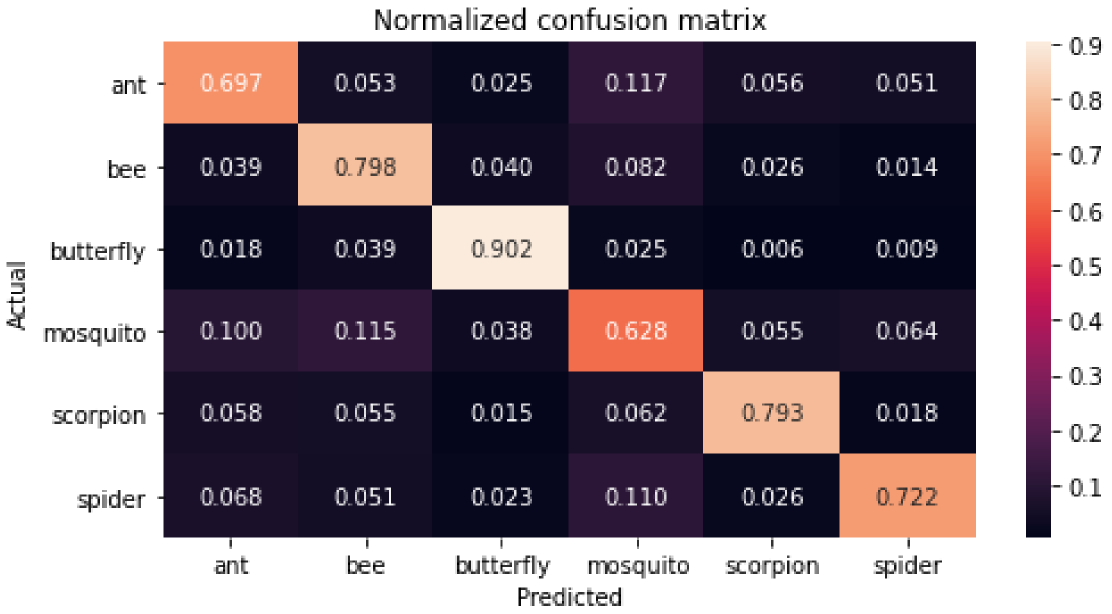


```
Test accuracy: 0.7571
```

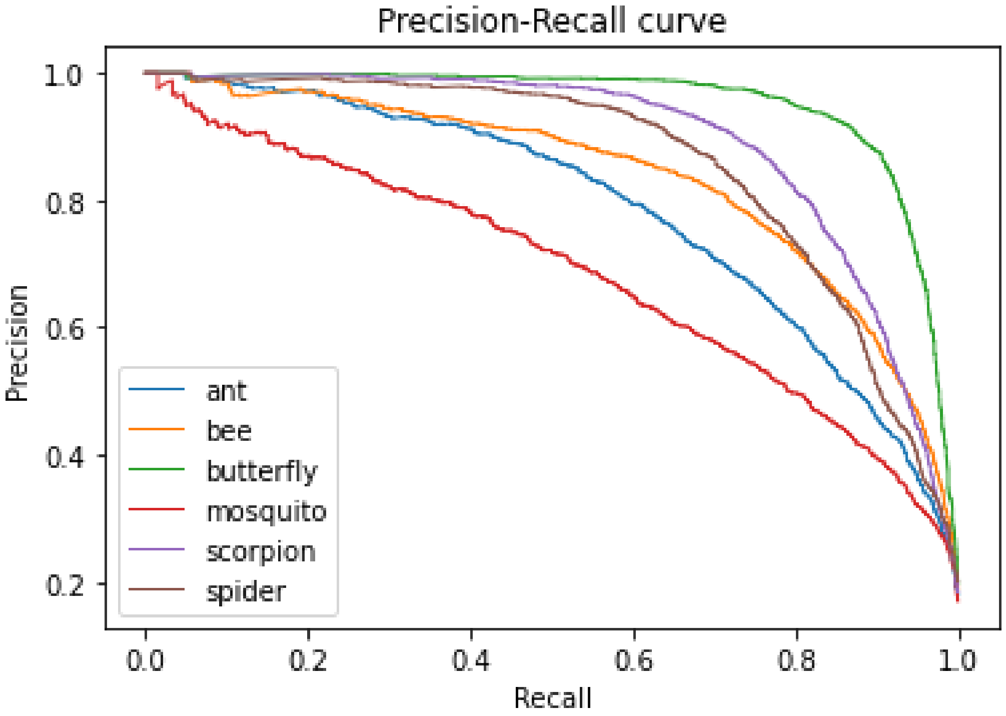


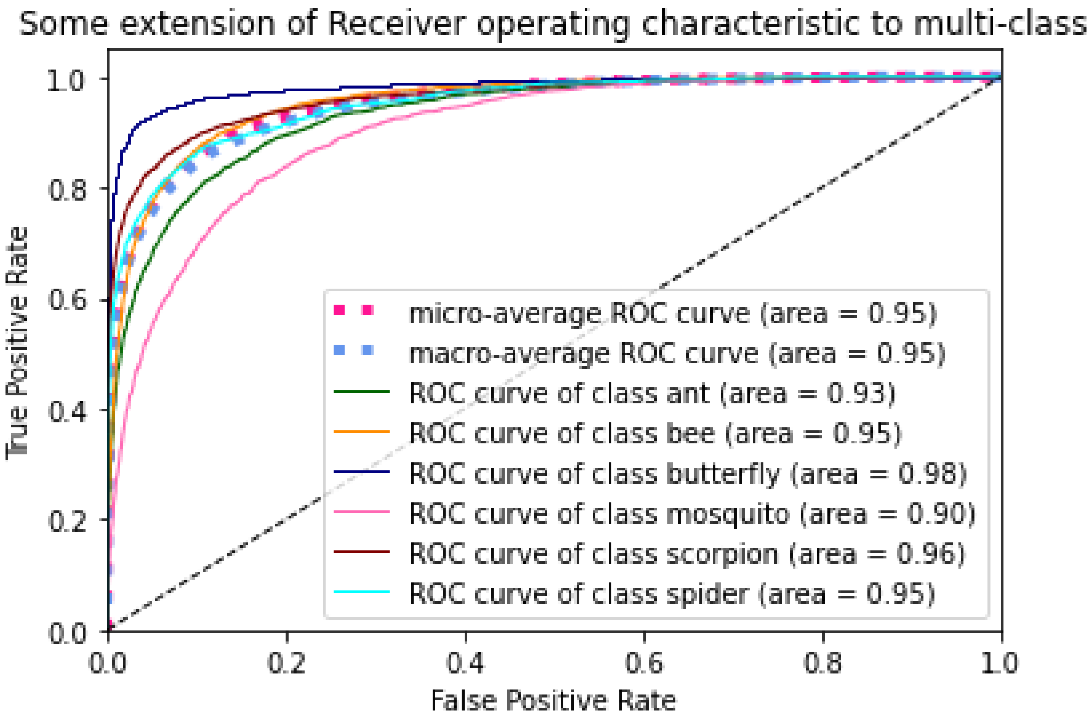
```
Test loss: 0.6967
```

```
              precision    recall  f1-score   support

         ant       0.71      0.70      0.70      2478
         bee       0.72      0.80      0.75      2488
   butterfly       0.87      0.90      0.88      2557
    mosquito       0.62      0.63      0.62      2527
    scorpion       0.82      0.79      0.81      2468
      spider       0.82      0.72      0.77      2482

    accuracy                           0.76     15000
   macro avg       0.76      0.76      0.76     15000
weighted avg       0.76      0.76      0.76     15000
```



	ant	bee	butterfly	mosquito	scorpion	spider
ant	0.697	0.053	0.025	0.117	0.056	0.051
bee	0.039	0.798	0.040	0.082	0.026	0.014
butterfly	0.018	0.039	0.902	0.025	0.006	0.009
mosquito	0.100	0.115	0.038	0.628	0.055	0.064
scorpion	0.058	0.055	0.015	0.062	0.793	0.018
spider	0.068	0.051	0.023	0.110	0.026	0.722



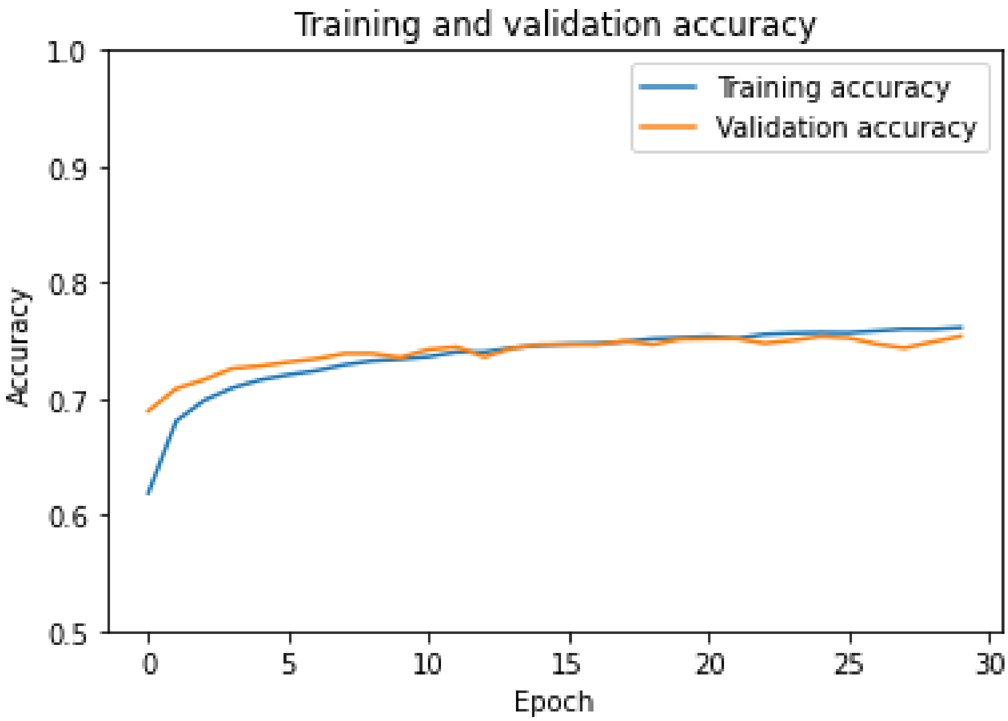


```
One-vs-One ROC AUC scores:
0.946891 (macro),
0.946948 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.946973 (macro),
0.947013 (weighted by prevalence)
```

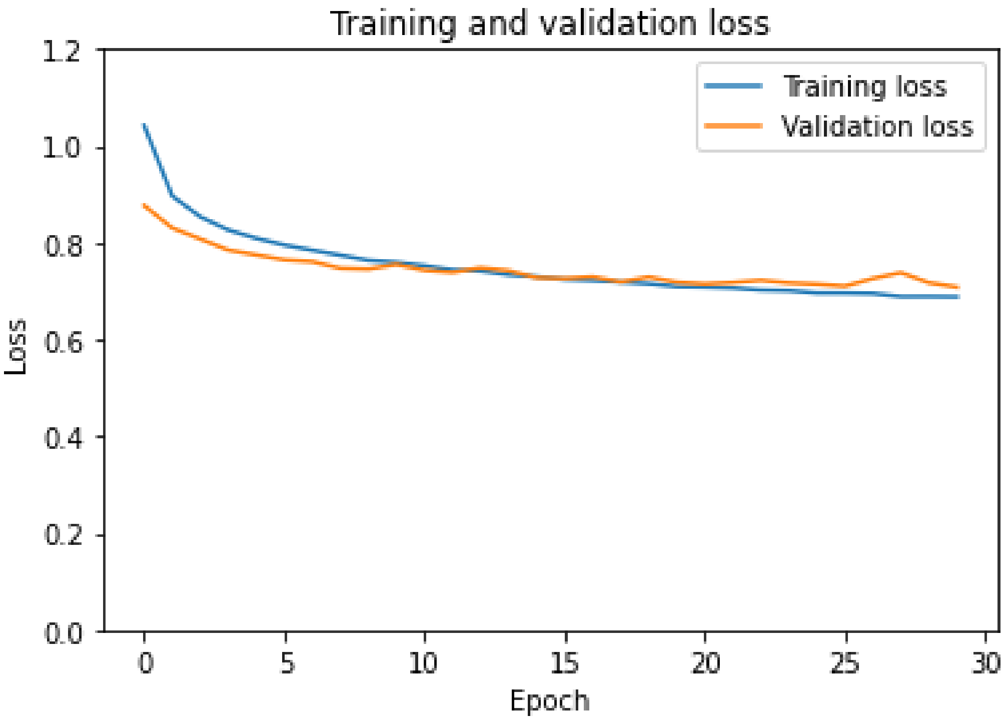
```
[ ]: model_20do, history_20do, score_20do, y_pred_20do = run_model(dropout=0.2)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.2DO_0L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

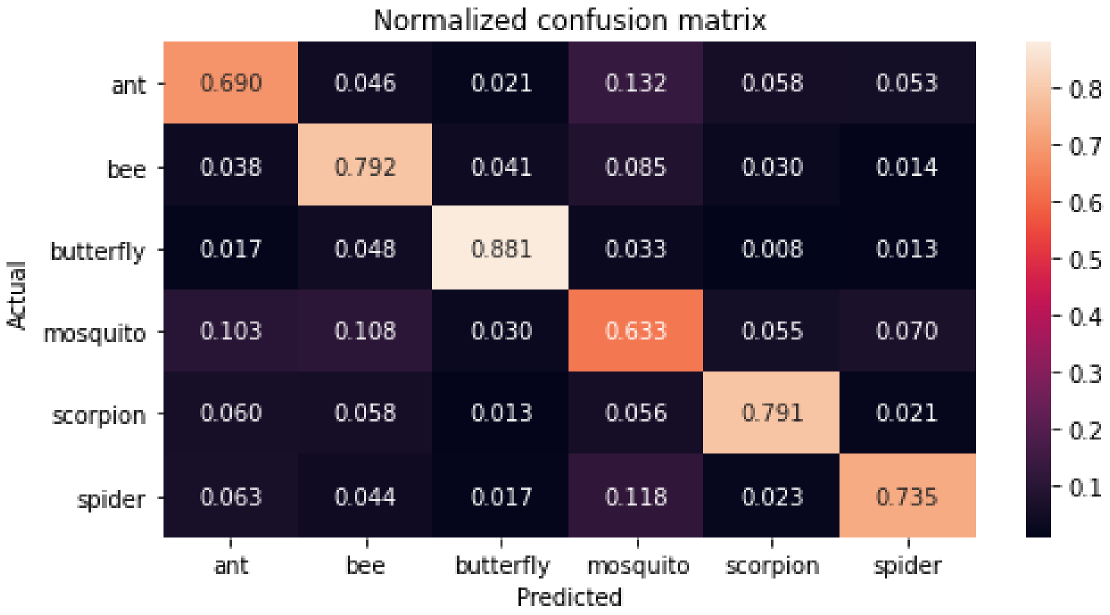
```
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120

_________________________________________________________________
BatchNormalization1 (BatchNo (None, 32)                128

_________________________________________________________________
Dense2_LeakyReLU (Dense)     (None, 32)                1056

_________________________________________________________________
Dropout_0.2 (Dropout)        (None, 32)                0

_________________________________________________________________
Dense3_LeakyReLU (Dense)     (None, 32)                1056

_________________________________________________________________
Dense4_LeakyReLU (Dense)     (None, 32)                1056

_________________________________________________________________
BatchNormalization2 (BatchNo (None, 32)                128

_________________________________________________________________
Dense5_LeakyReLU (Dense)     (None, 16)                528

_________________________________________________________________
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128

_________________________________________________________________
Total train time for 30 epochs = 71.239 seconds
```



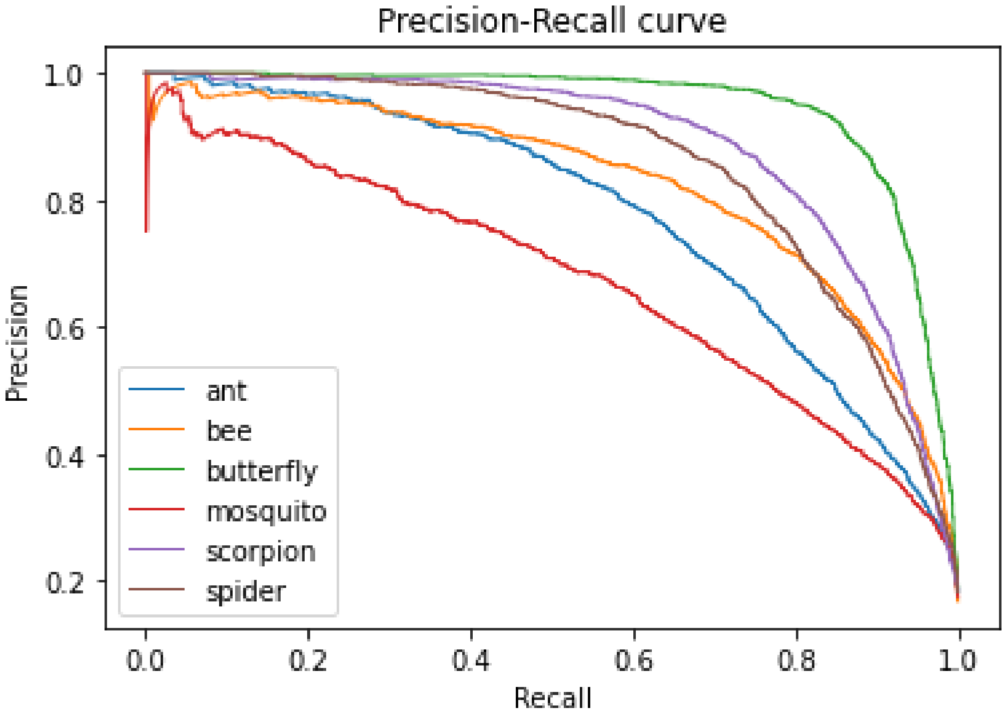
```
Test accuracy: 0.7538
```

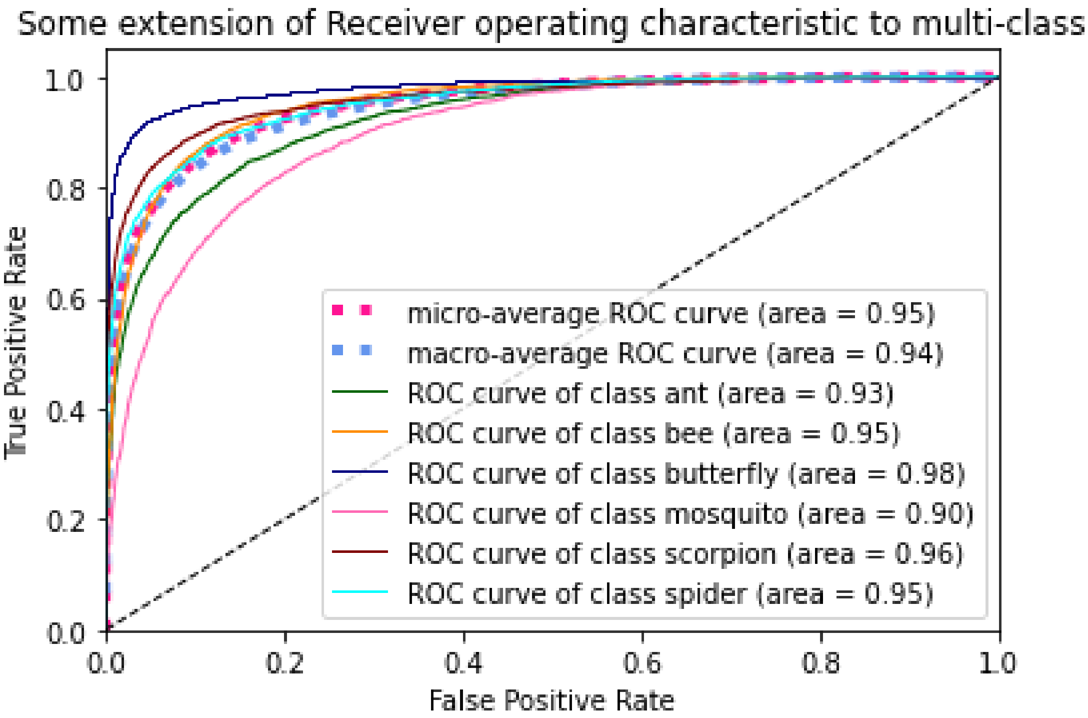


```
Test loss: 0.7123

              precision    recall  f1-score   support

         ant       0.71      0.69      0.70      2478
         bee       0.72      0.79      0.75      2488
   butterfly       0.88      0.88      0.88      2557
    mosquito       0.60      0.63      0.62      2527
    scorpion       0.82      0.79      0.80      2468
      spider       0.81      0.74      0.77      2482

    accuracy                           0.75     15000
   macro avg       0.76      0.75      0.75     15000
weighted avg       0.76      0.75      0.75     15000
```



	ant	bee	butterfly	mosquito	scorpion	spider
ant	0.690	0.046	0.021	0.132	0.058	0.053
bee	0.038	0.792	0.041	0.085	0.030	0.014
butterfly	0.017	0.048	0.881	0.033	0.008	0.013
mosquito	0.103	0.108	0.030	0.633	0.055	0.070
scorpion	0.060	0.058	0.013	0.056	0.791	0.021
spider	0.063	0.044	0.017	0.118	0.023	0.735



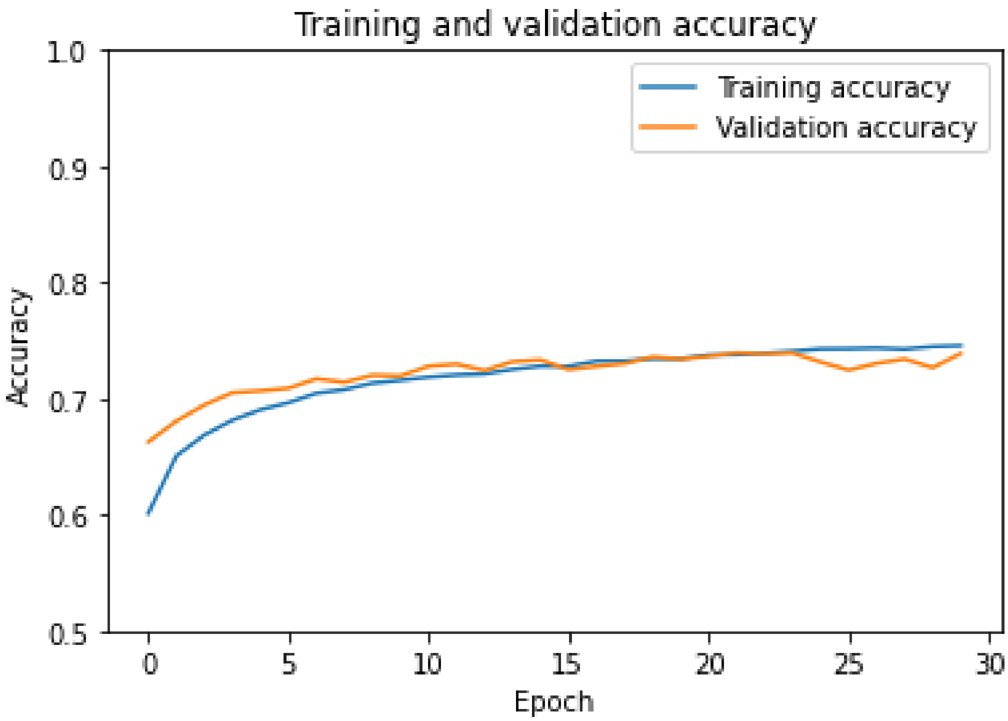


```
One-vs-One ROC AUC scores:
0.944132 (macro),
0.944187 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.944211 (macro),
0.944251 (weighted by prevalence)
```

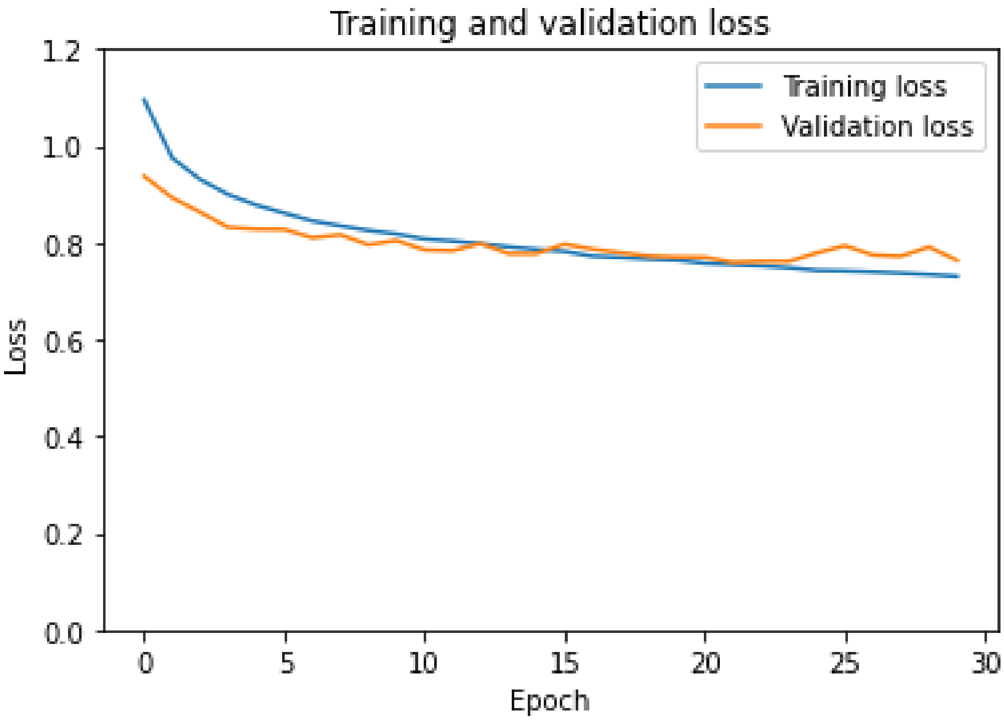
```
[ ]: model_40do, history_40do, score_40do, y_pred_40do = run_model(dropout=0.4)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.4DO_0L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

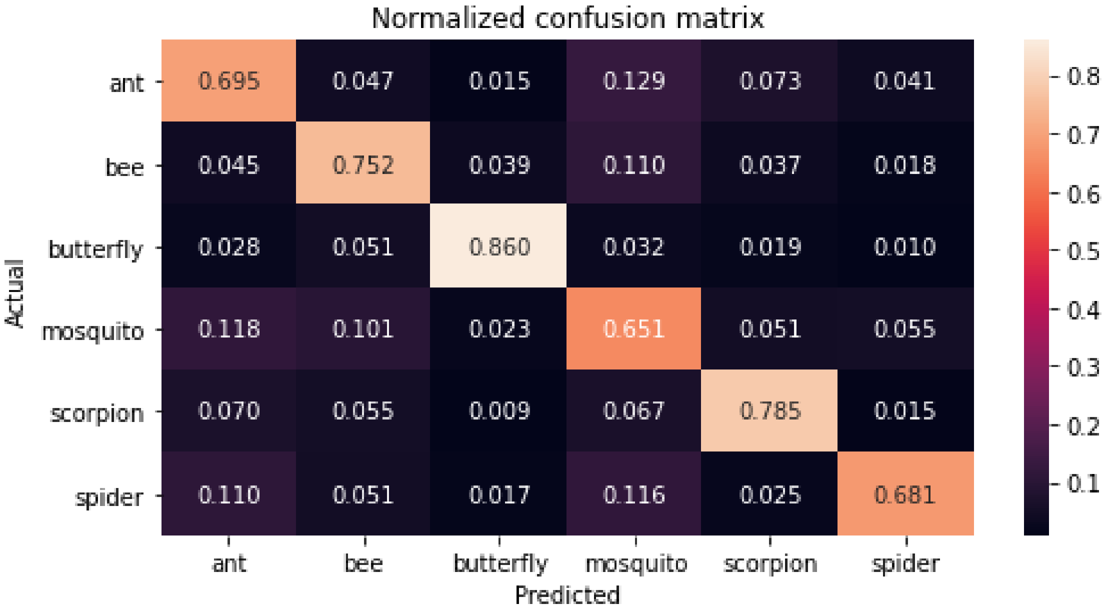
```
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120

-----------------------------------------------------------------
BatchNormalization1 (BatchNo (None, 32)                128

-----------------------------------------------------------------
Dense2_LeakyReLU (Dense)     (None, 32)                1056

-----------------------------------------------------------------
Dropout_0.4 (Dropout)        (None, 32)                0

-----------------------------------------------------------------
Dense3_LeakyReLU (Dense)     (None, 32)                1056

-----------------------------------------------------------------
Dense4_LeakyReLU (Dense)     (None, 32)                1056

-----------------------------------------------------------------
BatchNormalization2 (BatchNo (None, 32)                128

-----------------------------------------------------------------
Dense5_LeakyReLU (Dense)     (None, 16)                528

-----------------------------------------------------------------
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128

-----------------------------------------------------------------
Total train time for 30 epochs = 71.481 seconds
```



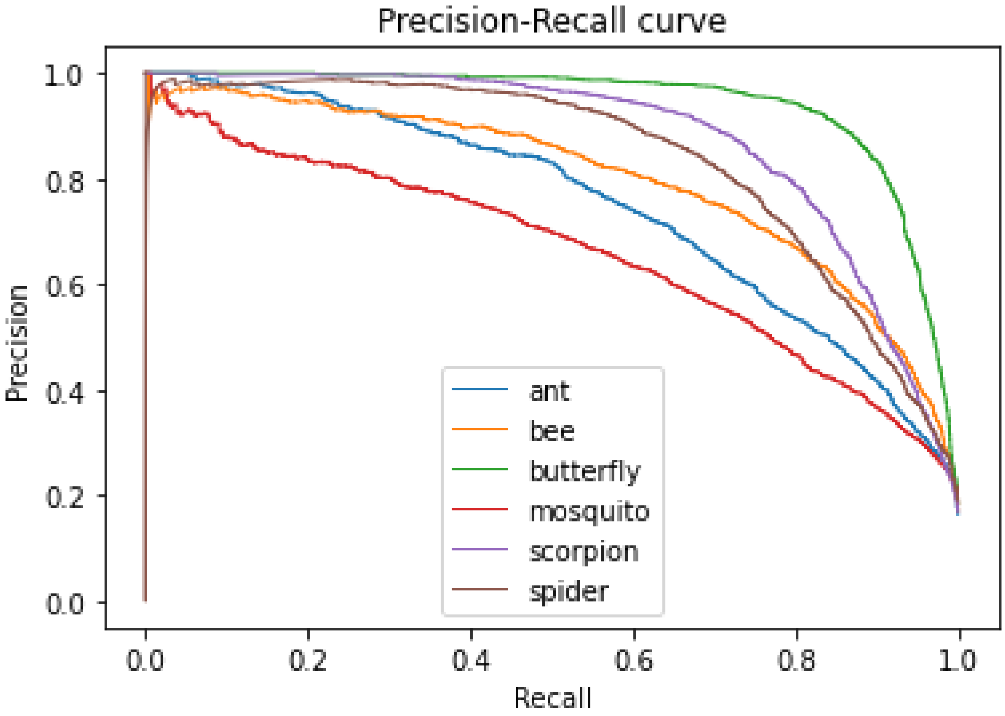
```
Test accuracy: 0.7375
```

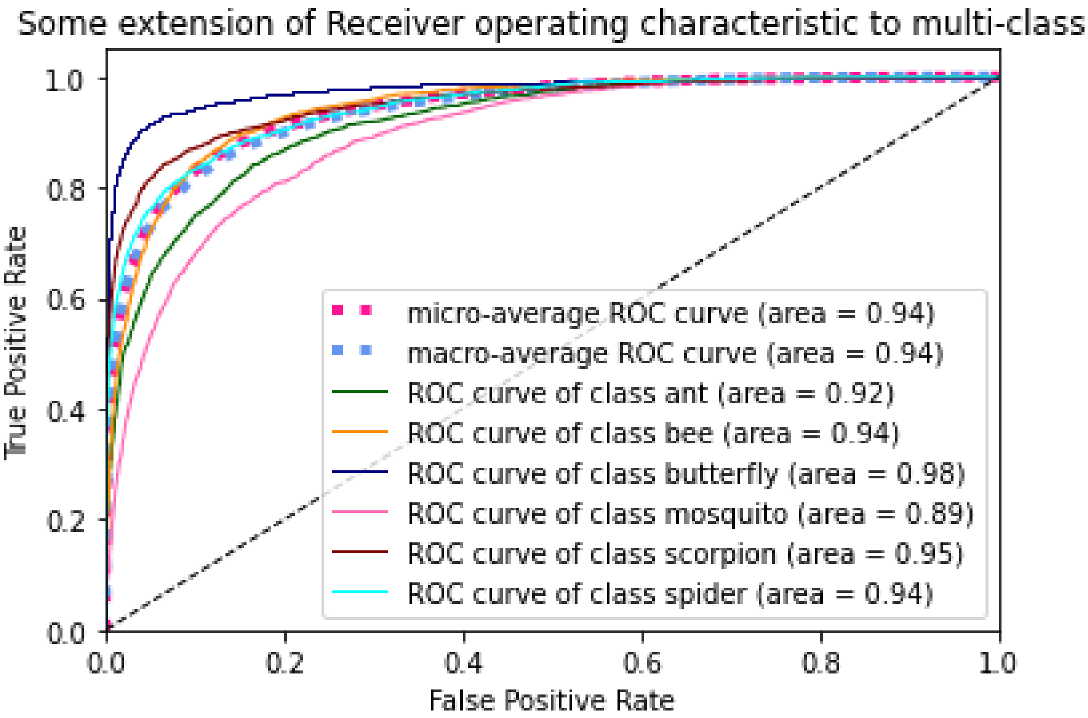


```
Test loss: 0.7637

              precision    recall  f1-score   support

         ant       0.65      0.69      0.67      2478
         bee       0.71      0.75      0.73      2488
   butterfly       0.90      0.86      0.88      2557
    mosquito       0.59      0.65      0.62      2527
    scorpion       0.79      0.78      0.79      2468
      spider       0.83      0.68      0.75      2482

    accuracy                           0.74     15000
   macro avg       0.74      0.74      0.74     15000
weighted avg       0.74      0.74      0.74     15000
```



Actual \ Predicted	ant	bee	butterfly	mosquito	scorpion	spider
ant	0.695	0.047	0.015	0.129	0.073	0.041
bee	0.045	0.752	0.039	0.110	0.037	0.018
butterfly	0.028	0.051	0.860	0.032	0.019	0.010
mosquito	0.118	0.101	0.023	0.651	0.051	0.055
scorpion	0.070	0.055	0.009	0.067	0.785	0.015
spider	0.110	0.051	0.017	0.116	0.025	0.681



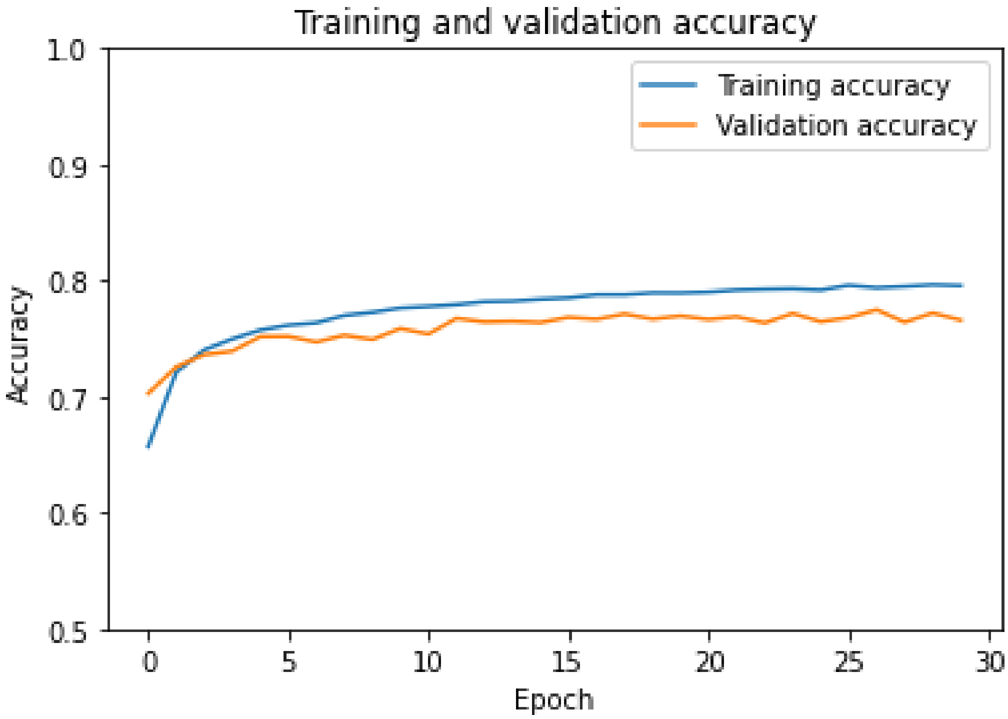


```
One-vs-One ROC AUC scores:
0.938007 (macro),
0.938062 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.938068 (macro),
0.938130 (weighted by prevalence)
```

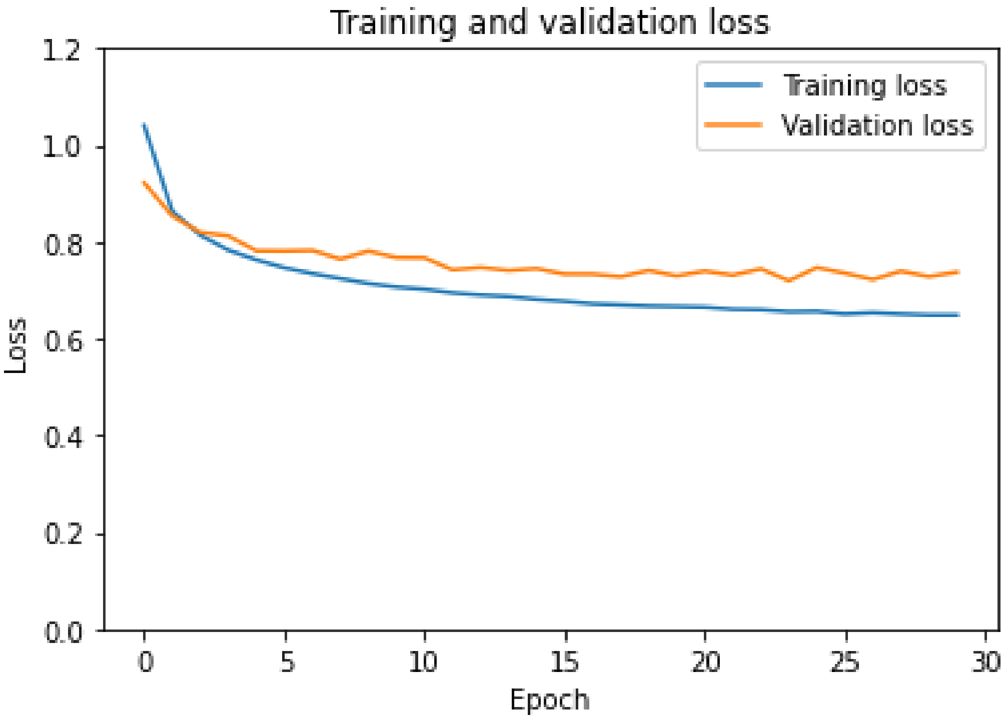
```
[ ]: model_0005L2, history_0005L2, score_0005L2, y_pred_0005L2 = run_model(l2_val=0.
     ↪0005)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0DO_0.0005L2"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120

-----------------------------------------------------------------
BatchNormalization1 (BatchNo (None, 32)                128

-----------------------------------------------------------------
Dense2_LeakyReLU (Dense)     (None, 32)                1056

-----------------------------------------------------------------
Dropout_0 (Dropout)          (None, 32)                0

-----------------------------------------------------------------
Dense3_LeakyReLU (Dense)     (None, 32)                1056

-----------------------------------------------------------------
Dense4_LeakyReLU (Dense)     (None, 32)                1056

-----------------------------------------------------------------
BatchNormalization2 (BatchNo (None, 32)                128

-----------------------------------------------------------------
Dense5_LeakyReLU (Dense)     (None, 16)                528

-----------------------------------------------------------------
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128

-----------------------------------------------------------------
Total train time for 30 epochs = 75.468 seconds
```

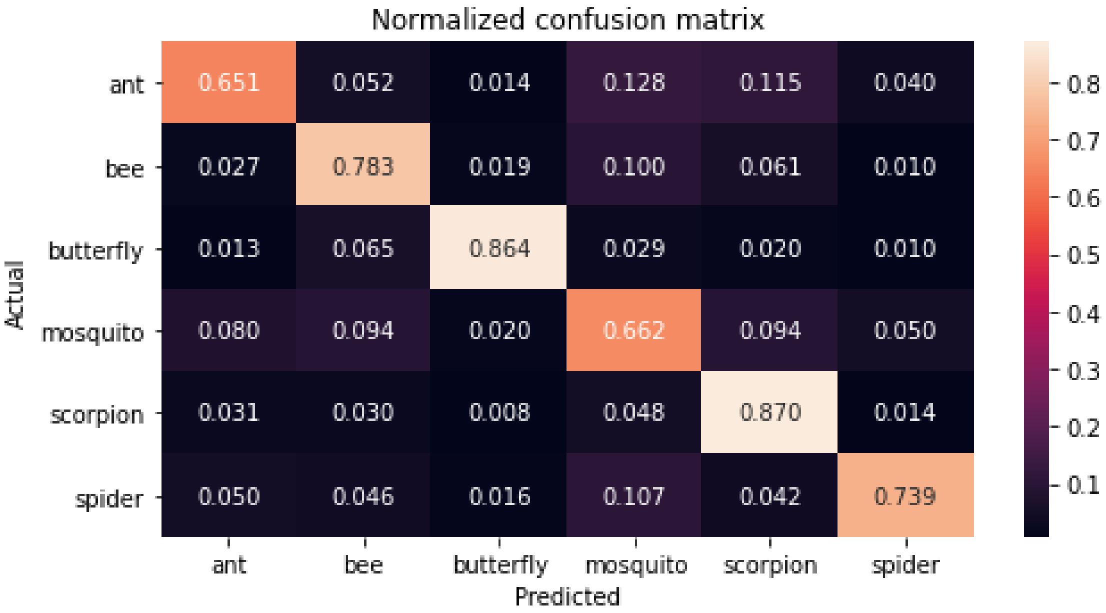


```
Test accuracy: 0.7613
```

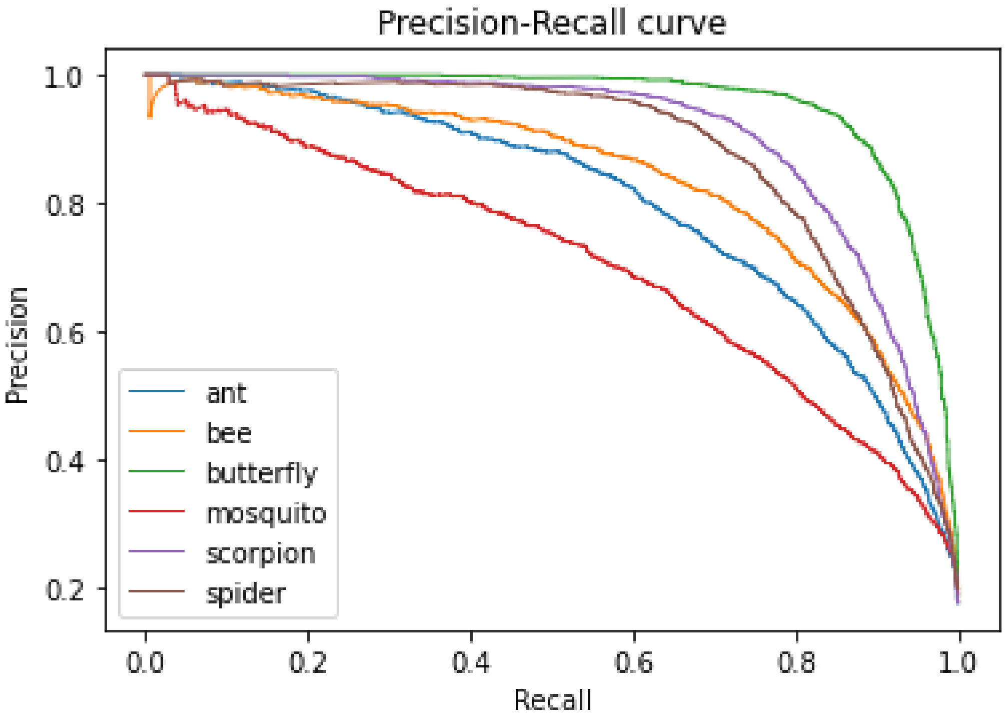


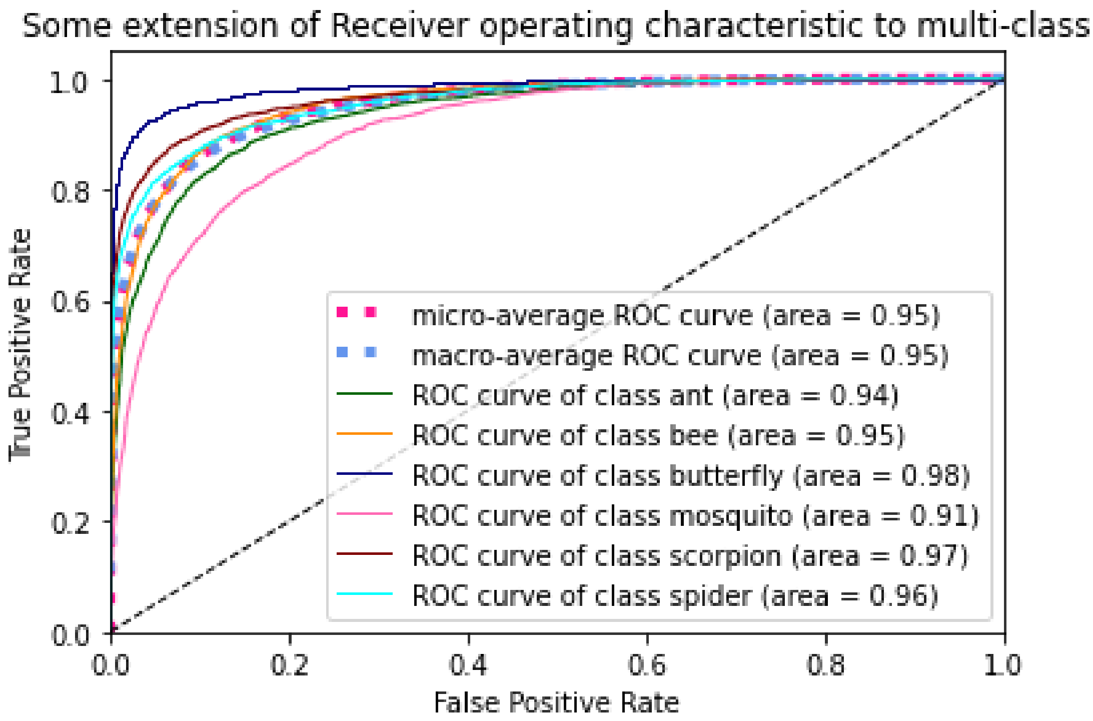
```
Test loss: 0.744
```

```
              precision    recall  f1-score   support

         ant       0.76      0.65      0.70      2478
         bee       0.73      0.78      0.76      2488
   butterfly       0.92      0.86      0.89      2557
    mosquito       0.62      0.66      0.64      2527
    scorpion       0.72      0.87      0.79      2468
      spider       0.86      0.74      0.79      2482

    accuracy                           0.76     15000
   macro avg       0.77      0.76      0.76     15000
weighted avg       0.77      0.76      0.76     15000
```



	ant	bee	butterfly	mosquito	scorpion	spider
ant	0.651	0.052	0.014	0.128	0.115	0.040
bee	0.027	0.783	0.019	0.100	0.061	0.010
butterfly	0.013	0.065	0.864	0.029	0.020	0.010
mosquito	0.080	0.094	0.020	0.662	0.094	0.050
scorpion	0.031	0.030	0.008	0.048	0.870	0.014
spider	0.050	0.046	0.016	0.107	0.042	0.739

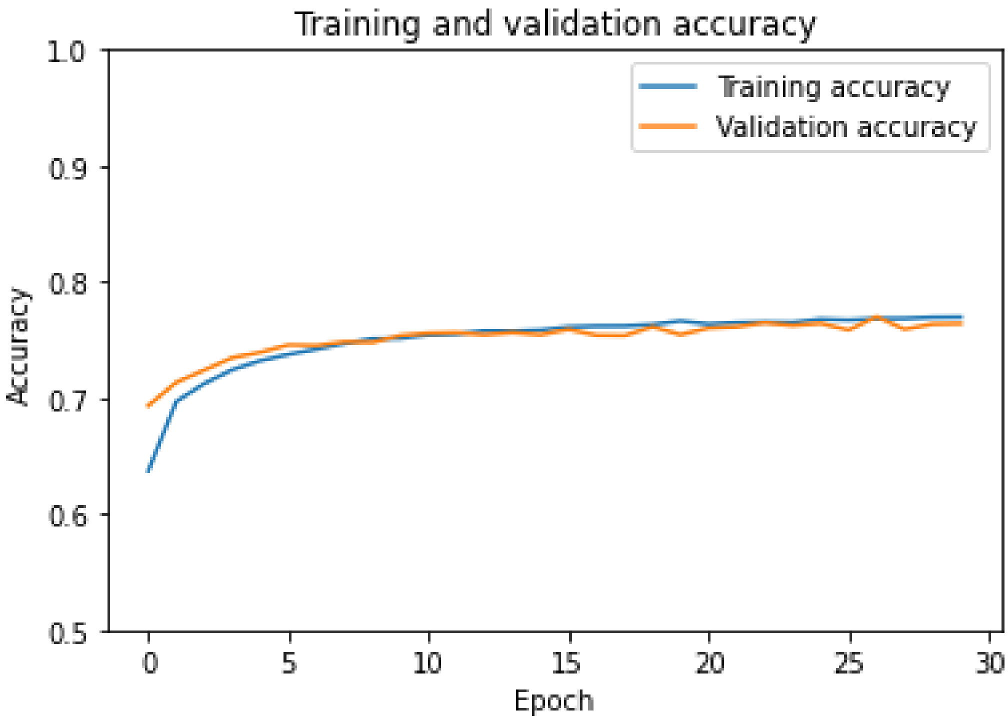




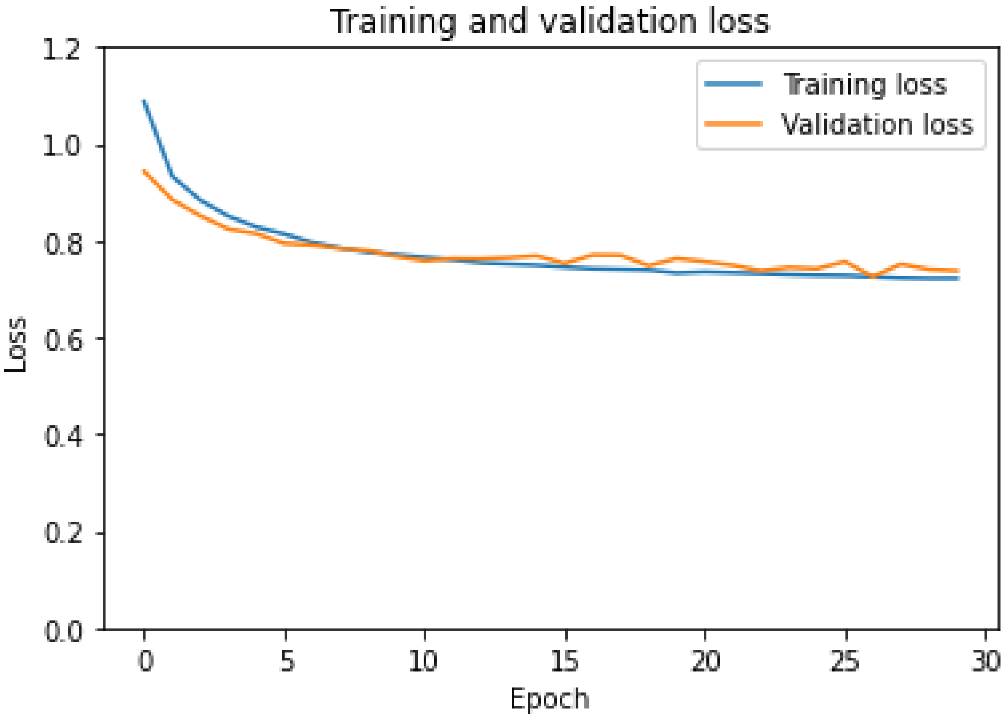
```
One-vs-One ROC AUC scores:
0.951254 (macro),
0.951290 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.951299 (macro),
0.951332 (weighted by prevalence)
```

```
[ ]: model_0005L2_10do, history_0005L2_10do, score_0005L2_10do, y_pred_0005L2_10do = run_model(l2_val=0.0005, dropout=0.1)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.1DO_0.0005L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120
_________________________________________________________________
BatchNormalization1 (BatchNo (None, 32)                128
_________________________________________________________________
Dense2_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dropout_0.1 (Dropout)        (None, 32)                0
_________________________________________________________________
Dense3_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dense4_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
BatchNormalization2 (BatchNo (None, 32)                128
_________________________________________________________________
Dense5_LeakyReLU (Dense)     (None, 16)                528
_________________________________________________________________
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128
_________________________________________________________________
Total train time for 30 epochs = 75.215 seconds
```

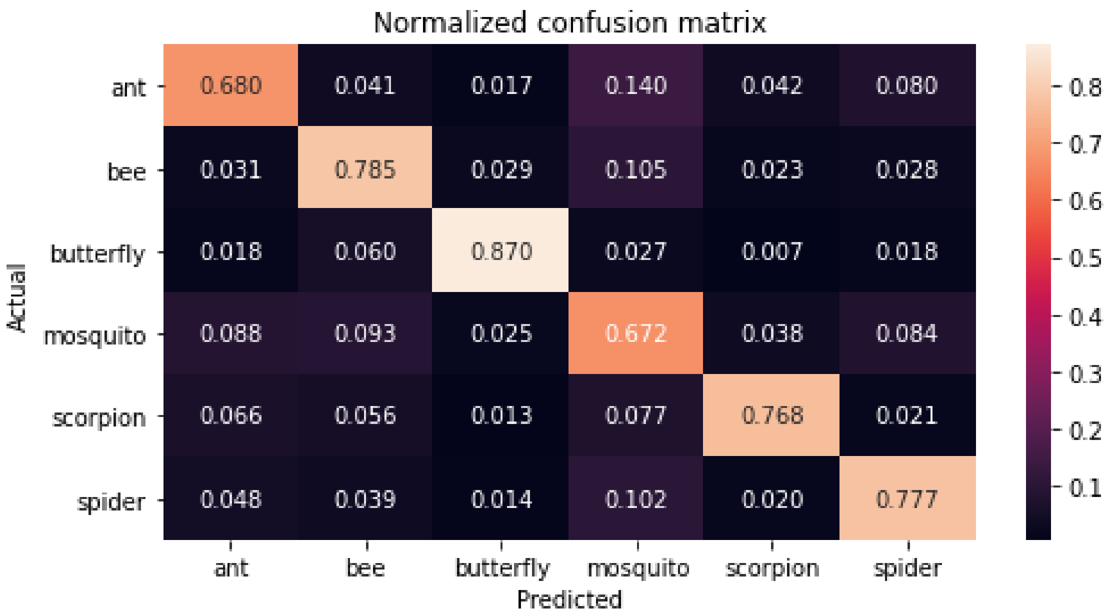


```
Test accuracy: 0.7588
```

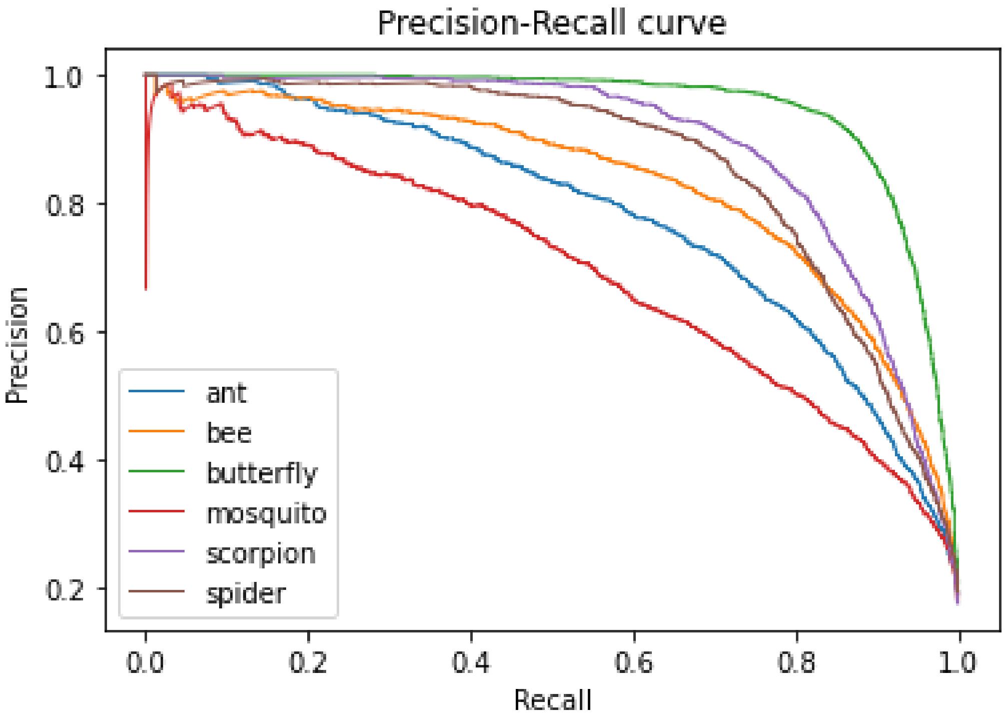


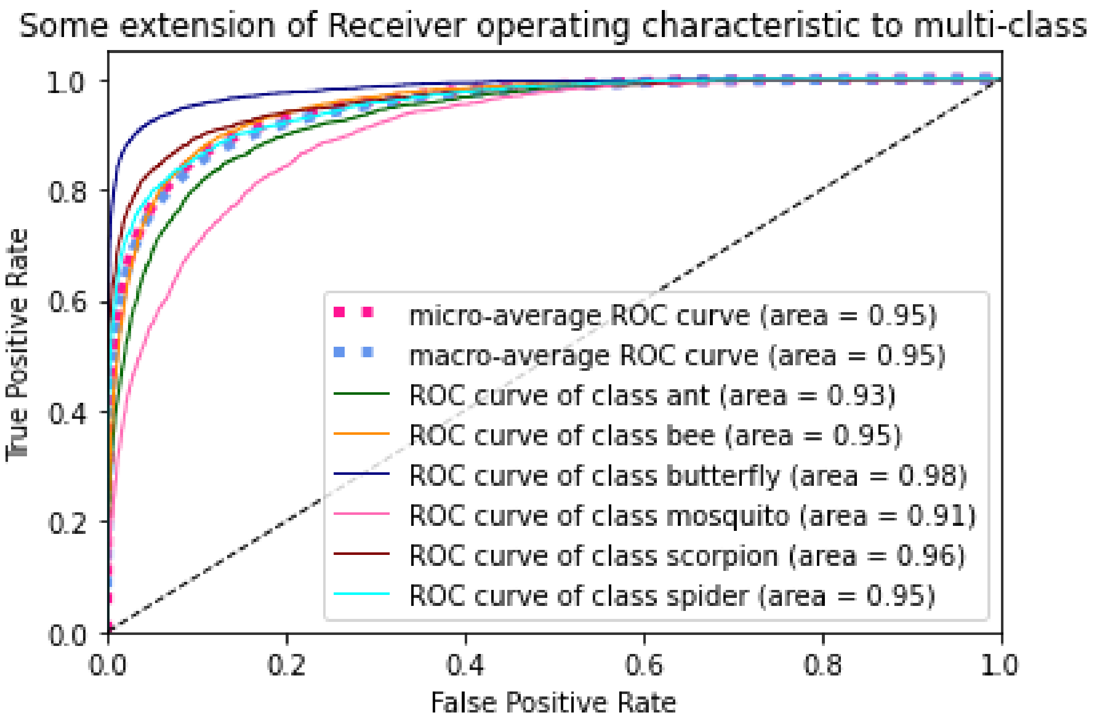
```
Test loss: 0.7538
```

```
              precision    recall  f1-score   support

         ant       0.73      0.68      0.70      2478
         bee       0.73      0.78      0.76      2488
   butterfly       0.90      0.87      0.89      2557
    mosquito       0.60      0.67      0.64      2527
    scorpion       0.85      0.77      0.81      2468
      spider       0.77      0.78      0.77      2482

    accuracy                           0.76     15000
   macro avg       0.76      0.76      0.76     15000
weighted avg       0.76      0.76      0.76     15000
```



Actual \ Predicted	ant	bee	butterfly	mosquito	scorpion	spider
ant	0.680	0.041	0.017	0.140	0.042	0.080
bee	0.031	0.785	0.029	0.105	0.023	0.028
butterfly	0.018	0.060	0.870	0.027	0.007	0.018
mosquito	0.088	0.093	0.025	0.672	0.038	0.084
scorpion	0.066	0.056	0.013	0.077	0.768	0.021
spider	0.048	0.039	0.014	0.102	0.020	0.777

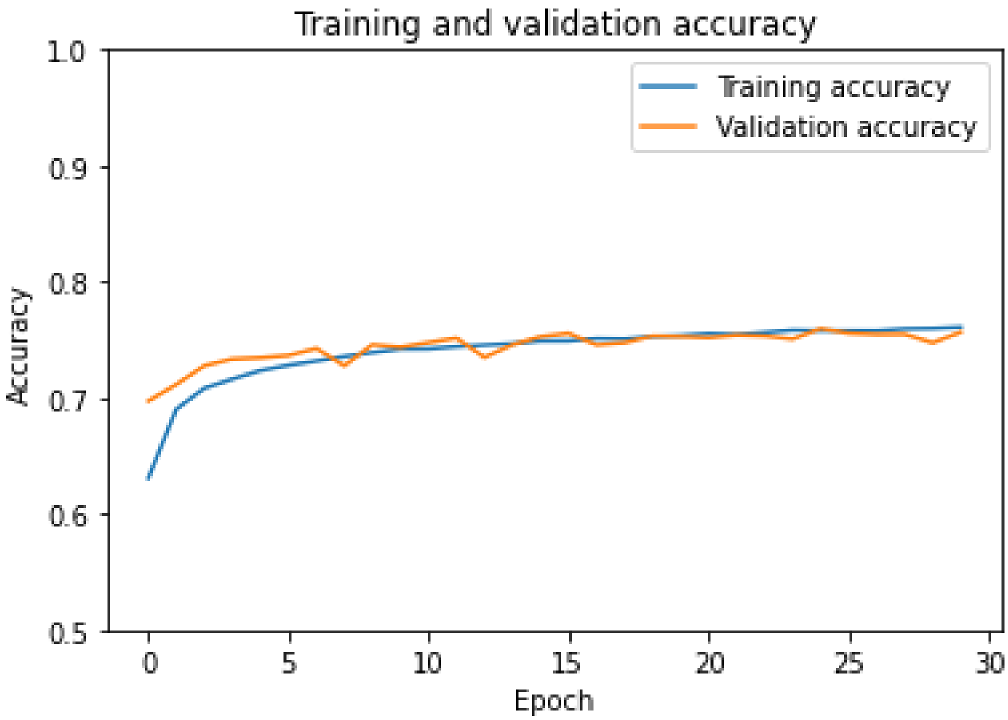




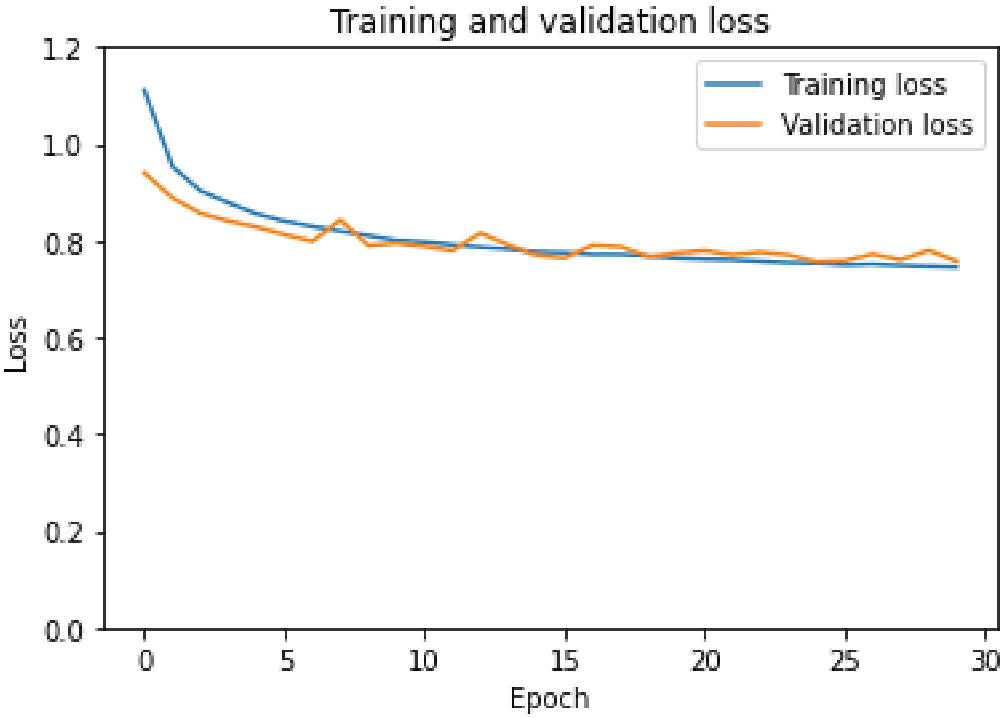
```
One-vs-One ROC AUC scores:
0.947766 (macro),
0.947825 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.947850 (macro),
0.947892 (weighted by prevalence)
```

[ ]:
```
model_0005L2_20do, history_0005L2_20do, score_0005L2_20do, y_pred_0005L2_20do =␣
 ↪run_model(l2_val=0.0005, dropout=0.2)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.2DO_0.0005L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120
-----------------------------------------------------------------
BatchNormalization1 (BatchNo (None, 32)                128
-----------------------------------------------------------------
Dense2_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
Dropout_0.2 (Dropout)        (None, 32)                0
-----------------------------------------------------------------
Dense3_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
Dense4_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
BatchNormalization2 (BatchNo (None, 32)                128
-----------------------------------------------------------------
Dense5_LeakyReLU (Dense)     (None, 16)                528
-----------------------------------------------------------------
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128
-----------------------------------------------------------------
Total train time for 30 epochs = 75.258 seconds
```

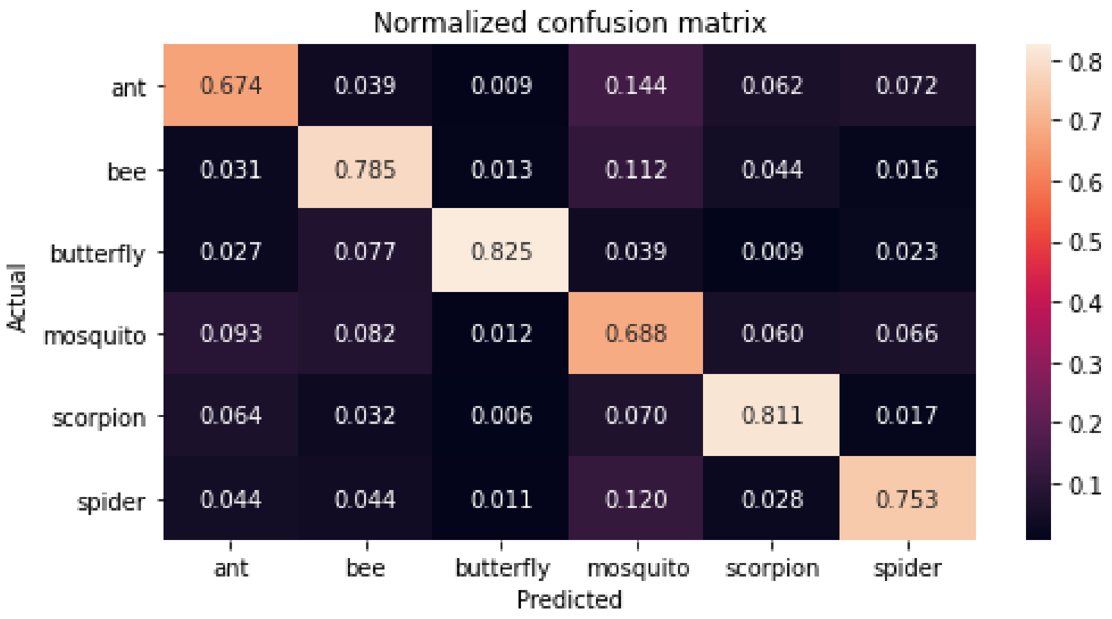


```
Test accuracy: 0.7559
```

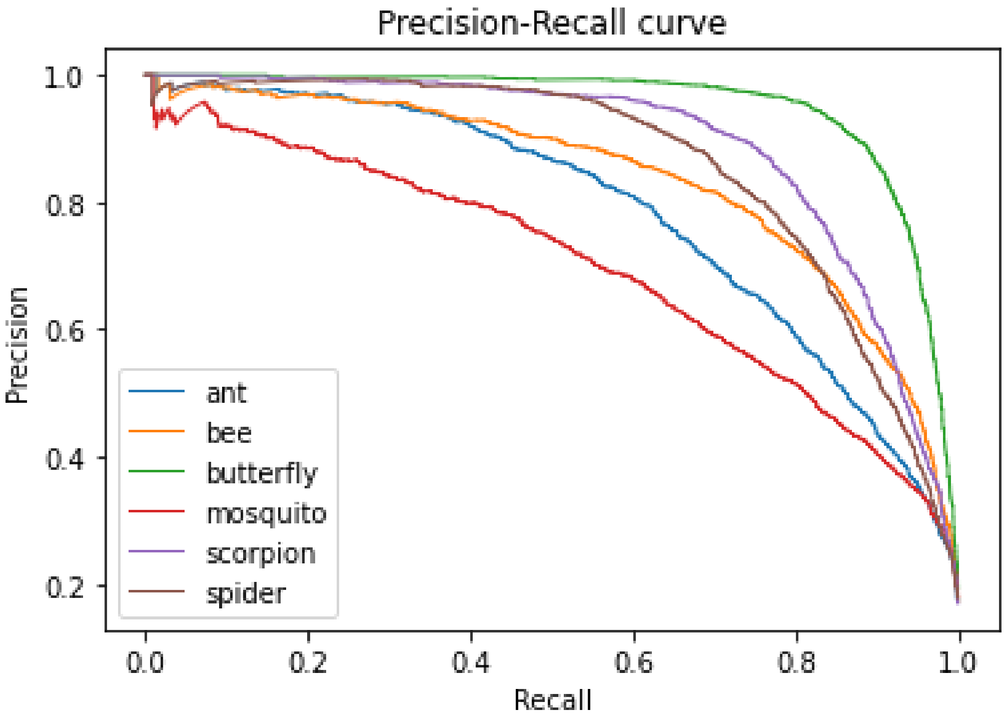


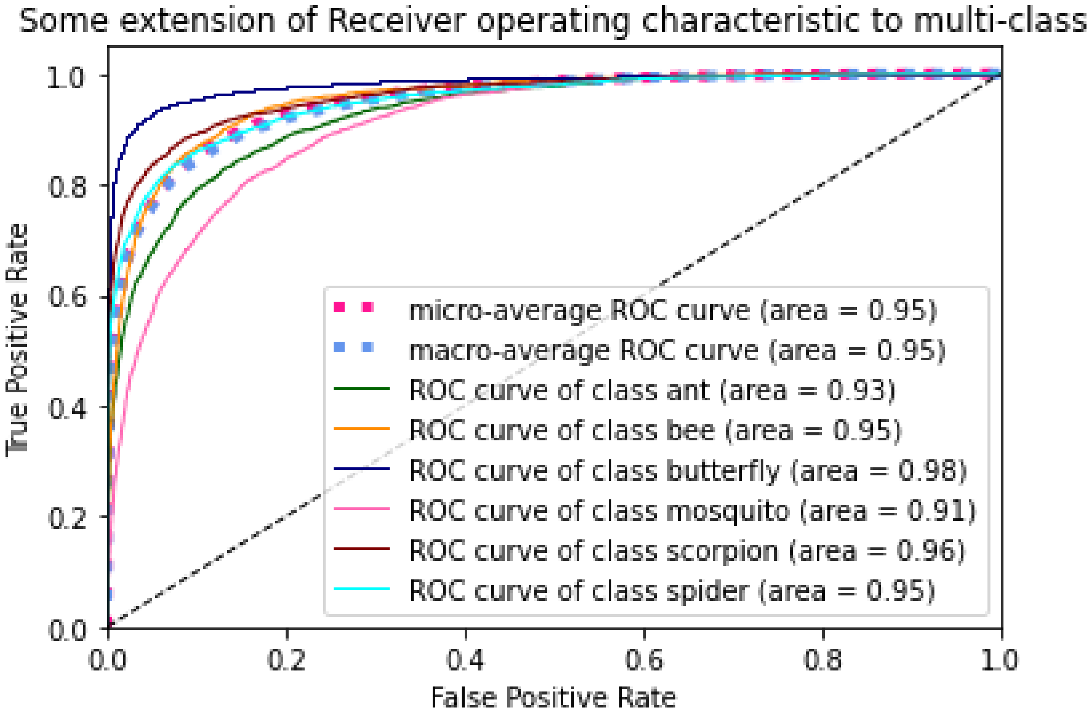
```
Test loss: 0.7577
```

```
              precision    recall  f1-score   support

         ant       0.72      0.67      0.70      2478
         bee       0.74      0.78      0.76      2488
   butterfly       0.94      0.82      0.88      2557
    mosquito       0.59      0.69      0.64      2527
    scorpion       0.80      0.81      0.80      2468
      spider       0.79      0.75      0.77      2482

    accuracy                           0.76     15000
   macro avg       0.76      0.76      0.76     15000
weighted avg       0.76      0.76      0.76     15000
```



	ant	bee	butterfly	mosquito	scorpion	spider
ant	0.674	0.039	0.009	0.144	0.062	0.072
bee	0.031	0.785	0.013	0.112	0.044	0.016
butterfly	0.027	0.077	0.825	0.039	0.009	0.023
mosquito	0.093	0.082	0.012	0.688	0.060	0.066
scorpion	0.064	0.032	0.006	0.070	0.811	0.017
spider	0.044	0.044	0.011	0.120	0.028	0.753

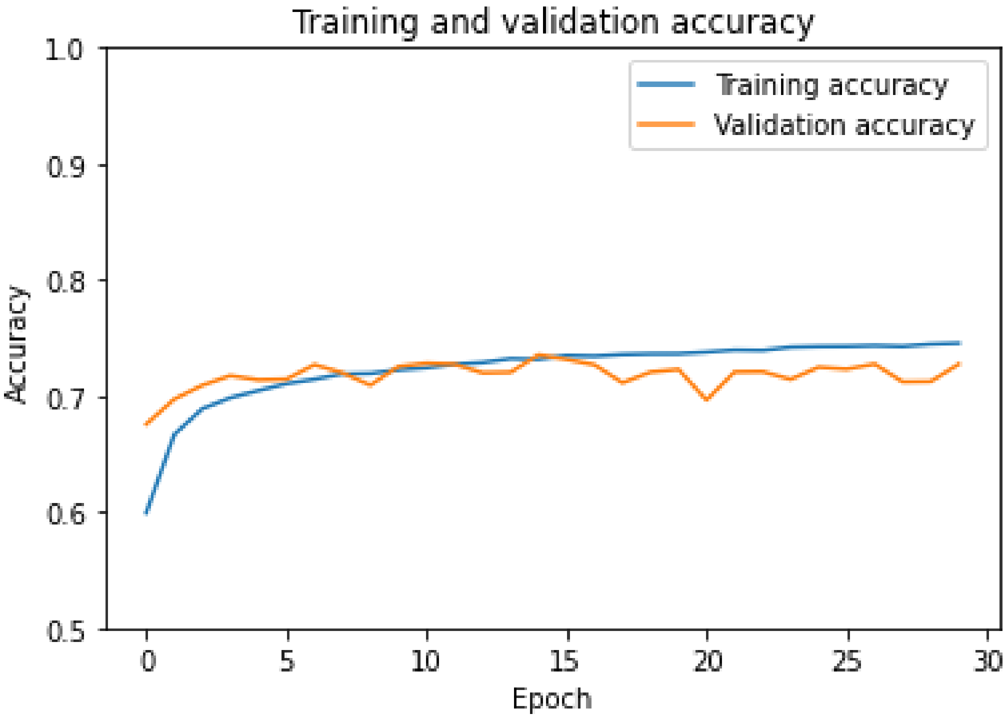




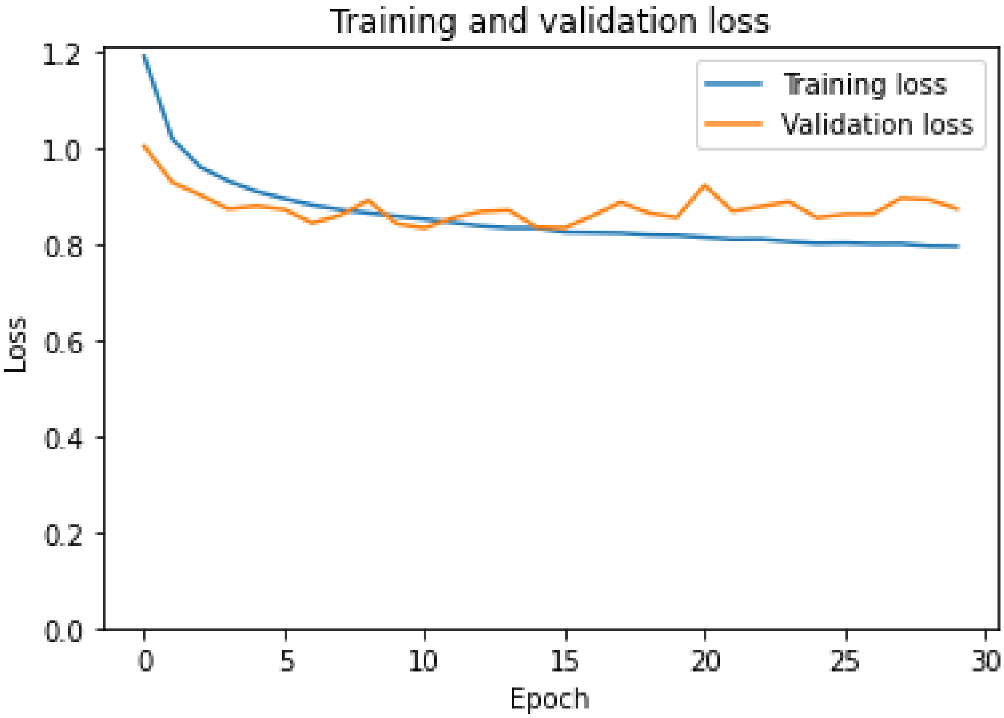
```
One-vs-One ROC AUC scores:
0.947502 (macro),
0.947534 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.947521 (macro),
0.947577 (weighted by prevalence)
```

[ ]:
```
model_0005L2_40do, history_0005L2_40do, score_0005L2_40do, y_pred_0005L2_40do =␣
 ↪run_model(l2_val=0.0005, dropout=0.4)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.4DO_0.0005L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120
-----------------------------------------------------------------
BatchNormalization1 (BatchNo (None, 32)                128
-----------------------------------------------------------------
Dense2_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
Dropout_0.4 (Dropout)        (None, 32)                0
-----------------------------------------------------------------
Dense3_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
Dense4_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
BatchNormalization2 (BatchNo (None, 32)                128
-----------------------------------------------------------------
Dense5_LeakyReLU (Dense)     (None, 16)                528
-----------------------------------------------------------------
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128
-----------------------------------------------------------------
Total train time for 30 epochs = 74.909 seconds
```

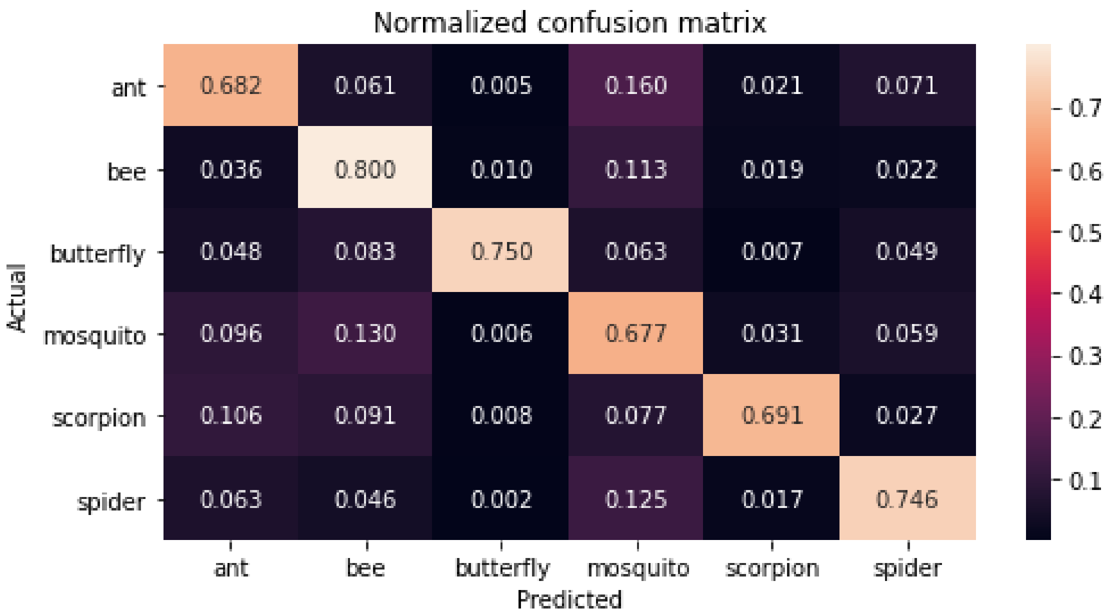


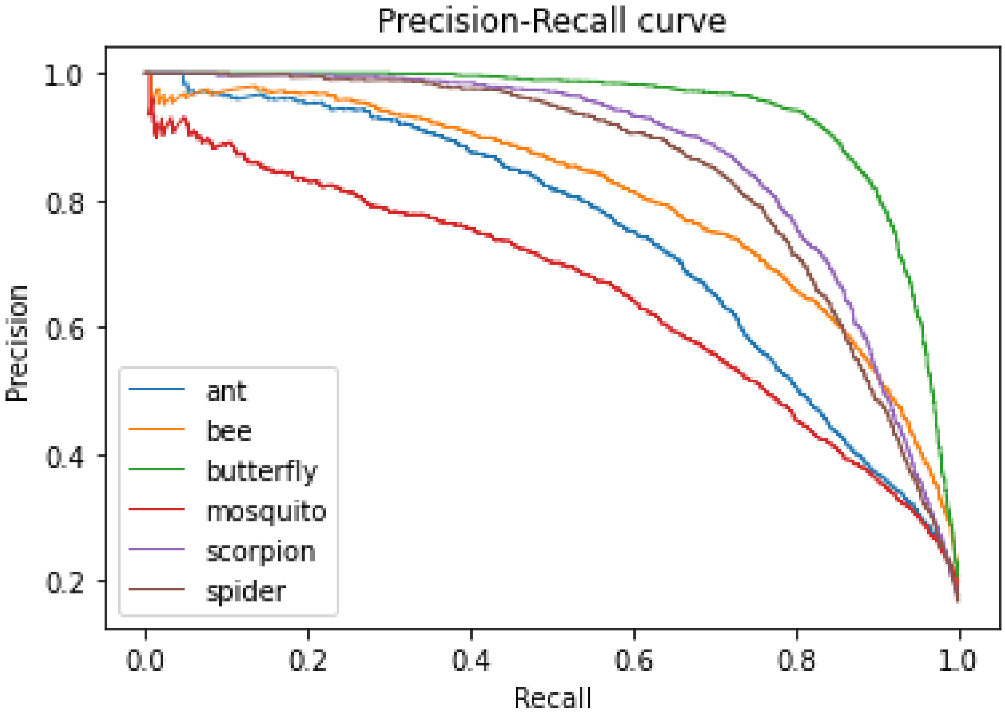
```
Test accuracy: 0.7245
```

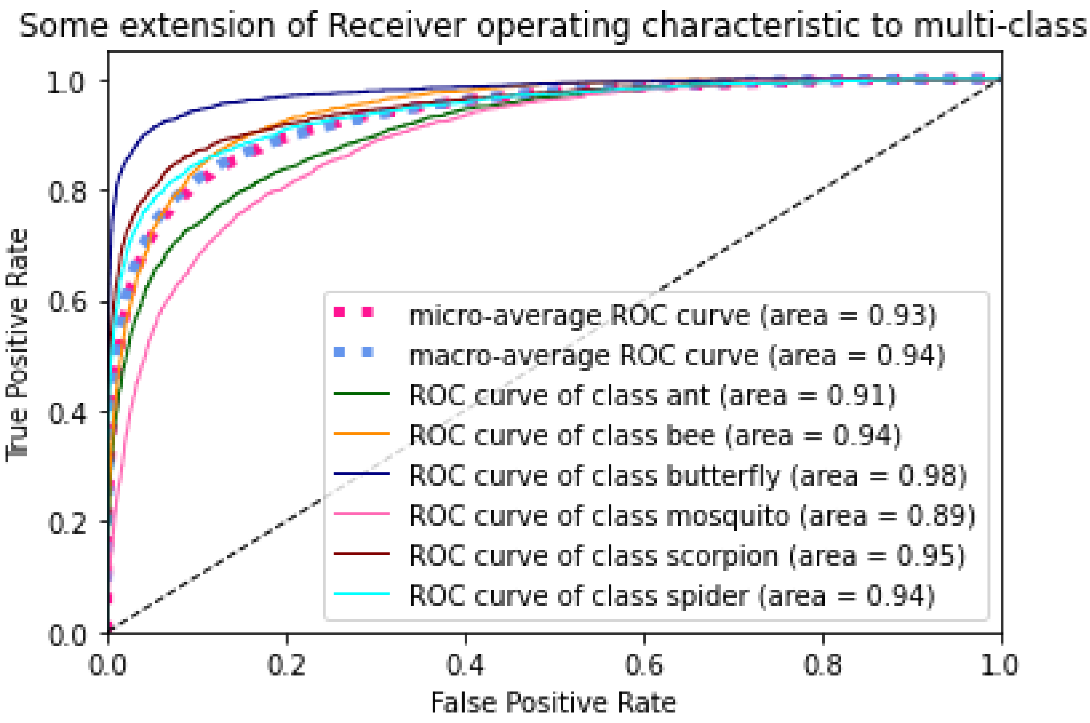


```
Test loss: 0.8783
```

```
              precision    recall  f1-score   support

         ant       0.66      0.68      0.67      2478
         bee       0.66      0.80      0.72      2488
   butterfly       0.96      0.75      0.84      2557
    mosquito       0.56      0.68      0.61      2527
    scorpion       0.88      0.69      0.77      2468
      spider       0.76      0.75      0.76      2482

    accuracy                           0.72     15000
   macro avg       0.75      0.72      0.73     15000
weighted avg       0.75      0.72      0.73     15000
```


Normalized confusion matrix
Actual
Predicted
ant 0.682 0.061 0.005 0.160 0.021 0.071
bee 0.036 0.800 0.010 0.113 0.019 0.022
butterfly 0.048 0.083 0.750 0.063 0.007 0.049
mosquito 0.096 0.130 0.006 0.677 0.031 0.059
scorpion 0.106 0.091 0.008 0.077 0.691 0.027
spider 0.063 0.046 0.002 0.125 0.017 0.746
ant
bee
butterfly
mosquito
scorpion
spider
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Precision-Recall curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
ant
bee
butterfly
mosquito
scorpion
spider

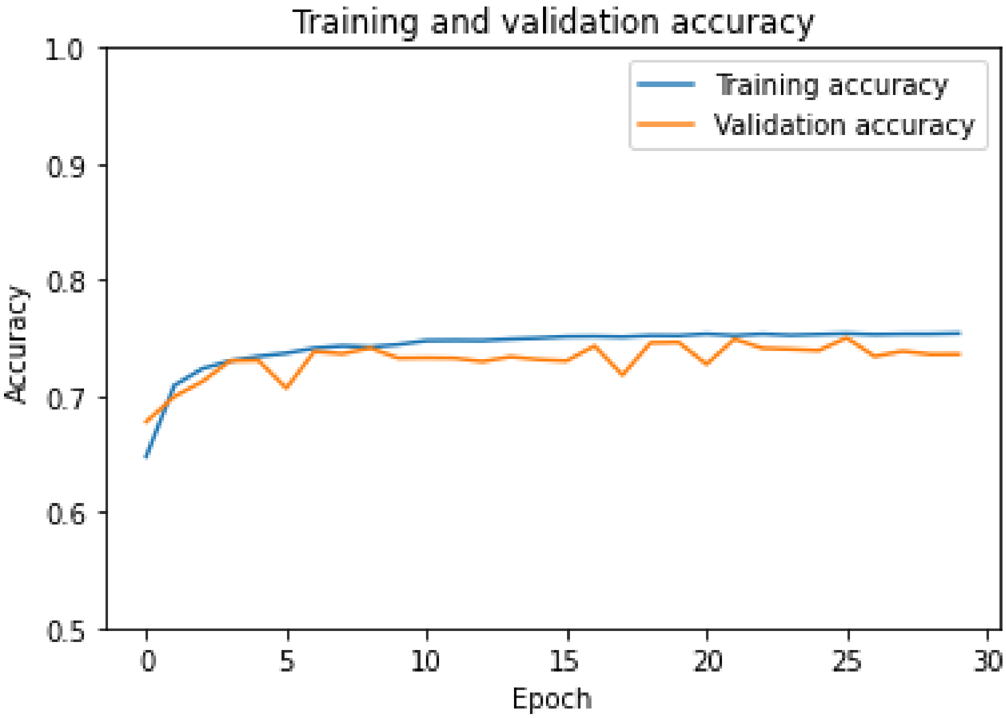


```
One-vs-One ROC AUC scores:
0.935029 (macro),
0.935063 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.935038 (macro),
0.935112 (weighted by prevalence)
```

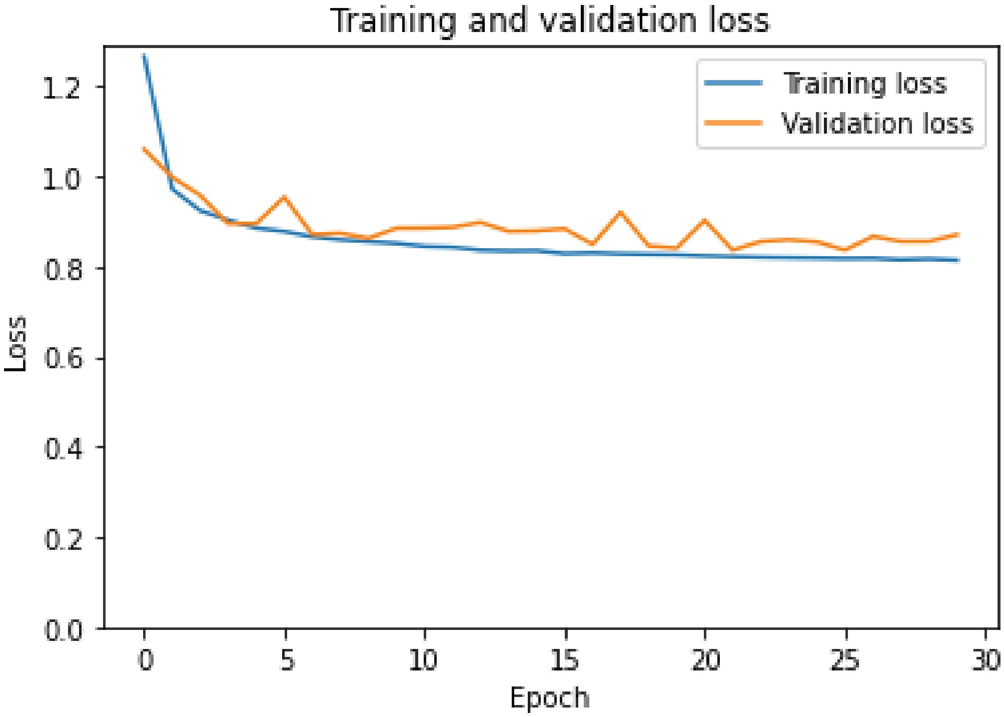
```
[ ]: model_01L2, history_01L2, score_01L2, y_pred_01L2 = run_model(l2_val=0.01)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0DO_0.01L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120
```

```
-----------------------------------------------------------------
BatchNormalization1 (BatchNo (None, 32)                128
-----------------------------------------------------------------
Dense2_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
Dropout_0 (Dropout)          (None, 32)                0
-----------------------------------------------------------------
Dense3_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
Dense4_LeakyReLU (Dense)     (None, 32)                1056
-----------------------------------------------------------------
BatchNormalization2 (BatchNo (None, 32)                128
-----------------------------------------------------------------
Dense5_LeakyReLU (Dense)     (None, 16)                528
-----------------------------------------------------------------
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128

-----------------------------------------------------------------
Total train time for 30 epochs = 75.891 seconds
```

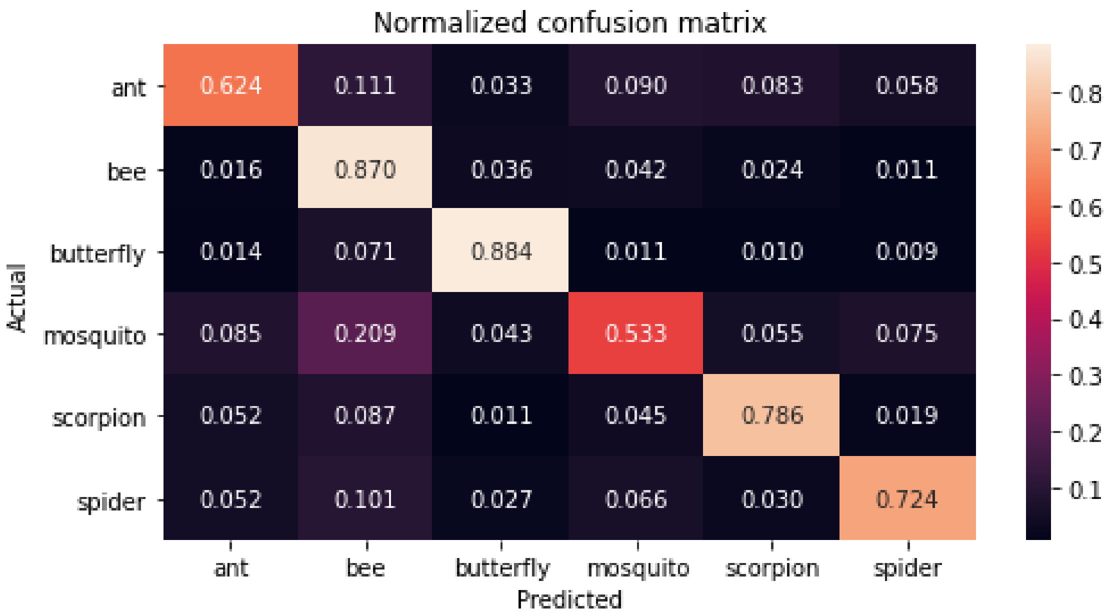


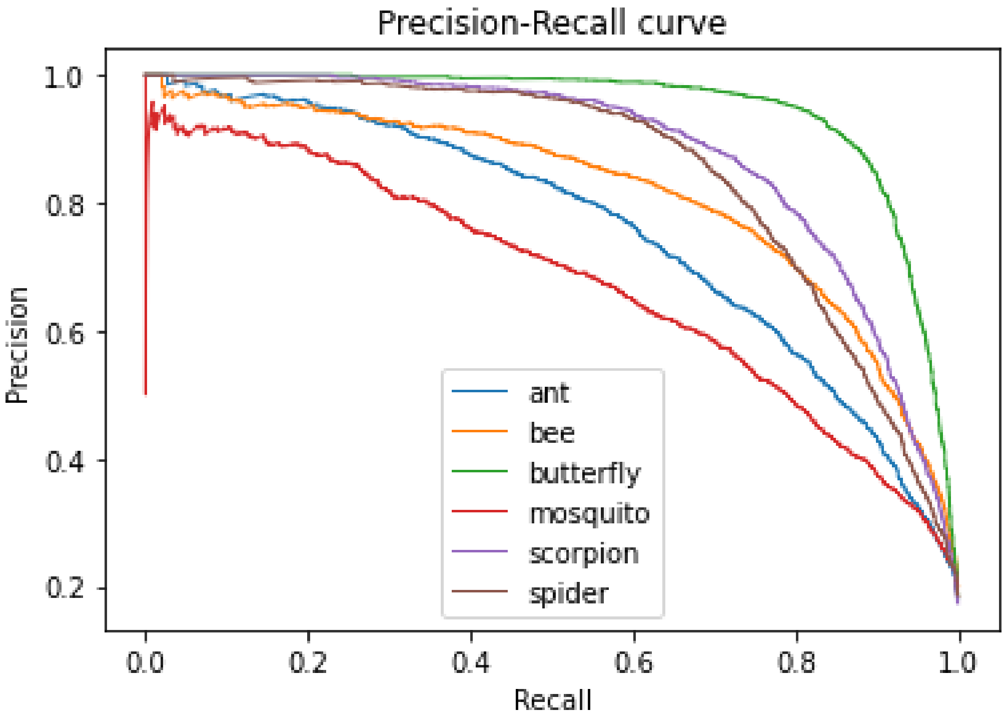
```
Test accuracy: 0.7371
```

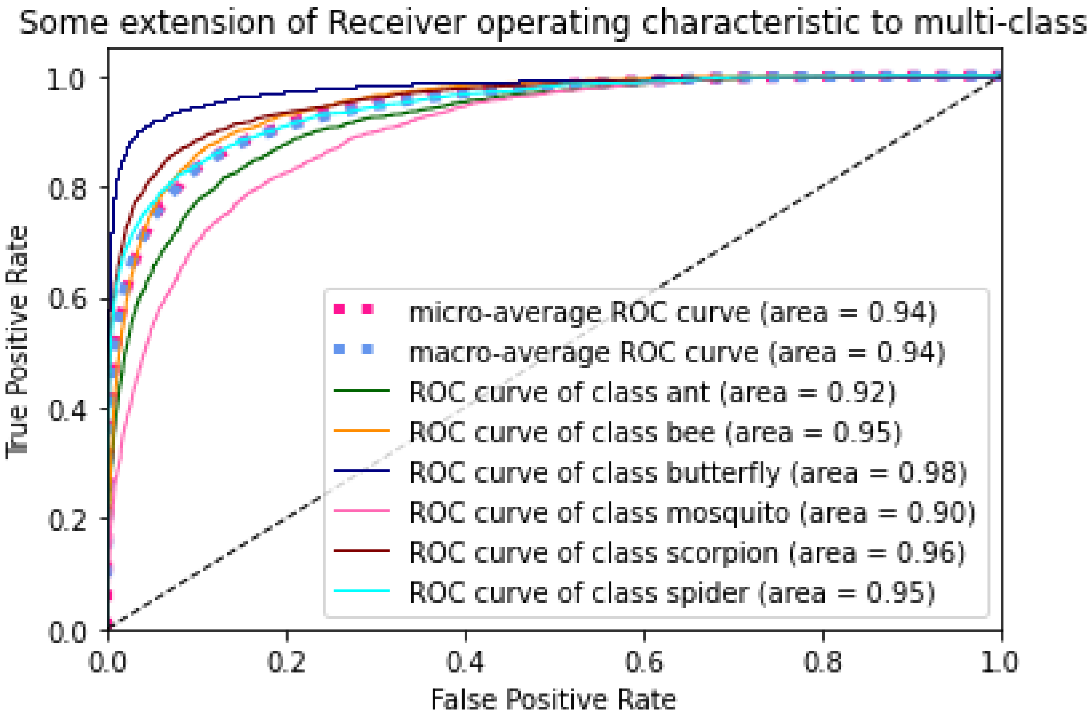


```
Test loss: 0.8738
```

```
              precision    recall  f1-score   support

         ant       0.74      0.62      0.68      2478
         bee       0.60      0.87      0.71      2488
   butterfly       0.86      0.88      0.87      2557
    mosquito       0.68      0.53      0.60      2527
    scorpion       0.79      0.79      0.79      2468
      spider       0.81      0.72      0.76      2482

    accuracy                           0.74     15000
   macro avg       0.75      0.74      0.73     15000
weighted avg       0.75      0.74      0.73     15000
```


Normalized confusion matrix
Actual
Predicted
ant 0.624 0.111 0.033 0.090 0.083 0.058
bee 0.016 0.870 0.036 0.042 0.024 0.011
butterfly 0.014 0.071 0.884 0.011 0.010 0.009
mosquito 0.085 0.209 0.043 0.533 0.055 0.075
scorpion 0.052 0.087 0.011 0.045 0.786 0.019
spider 0.052 0.101 0.027 0.066 0.030 0.724
ant
bee
butterfly
mosquito
scorpion
spider
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

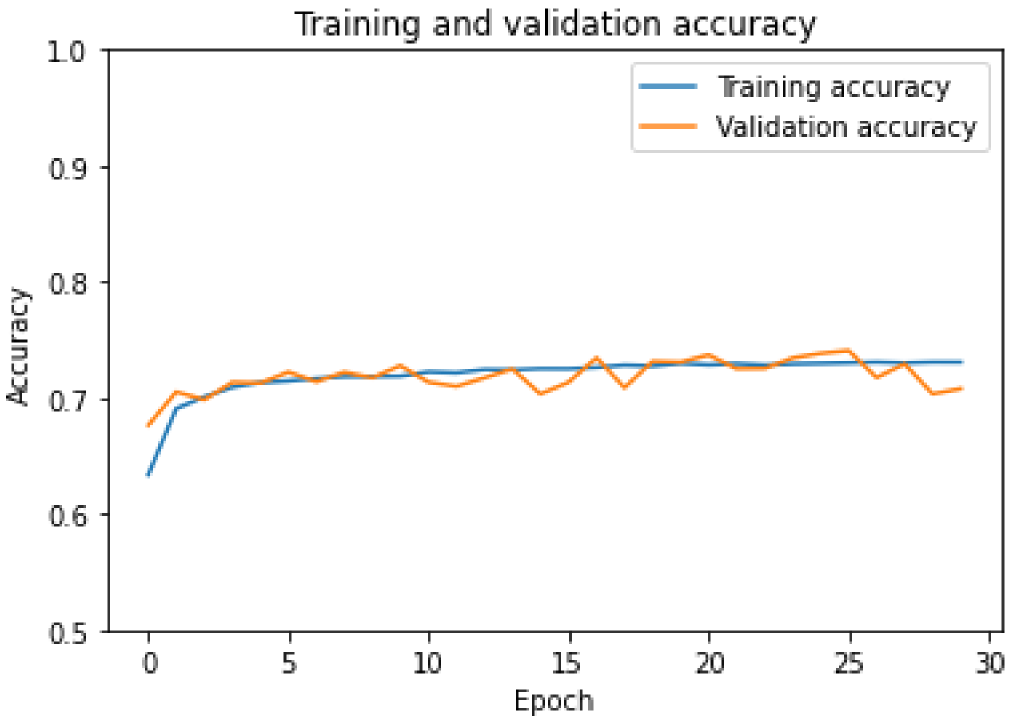

Precision-Recall curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
ant
bee
butterfly
mosquito
scorpion
spider



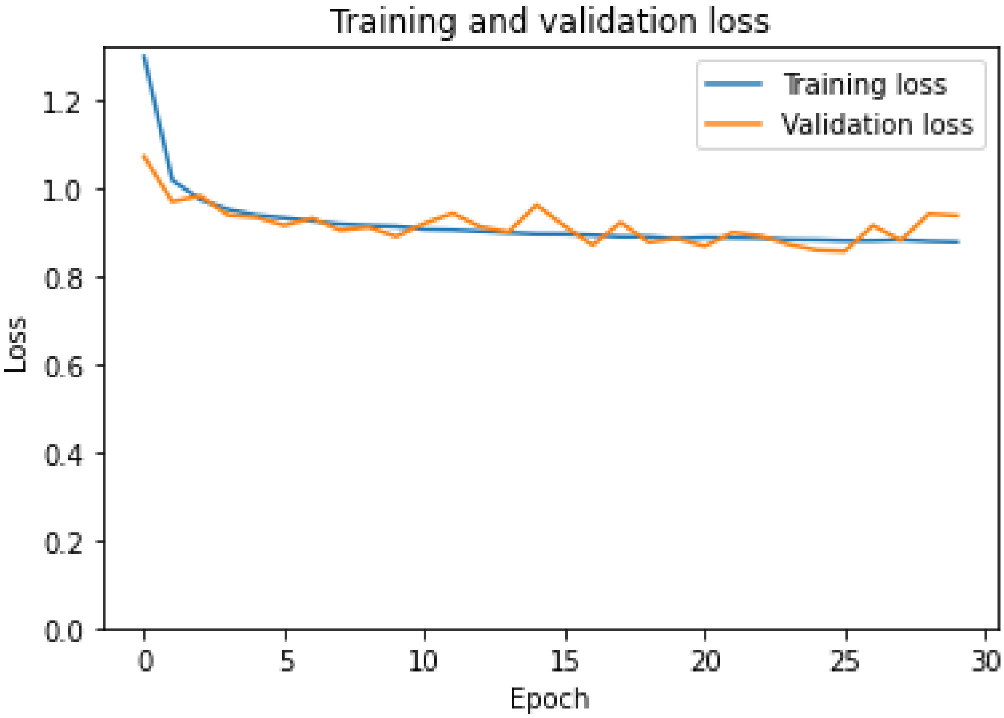
```
One-vs-One ROC AUC scores:
0.941489 (macro),
0.941559 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.941585 (macro),
0.941641 (weighted by prevalence)
```

```
[ ]: model_01L2_10do, history_01L2_10do, score_01L2_10do, y_pred_01L2_10do =␣
      ↪run_model(l2_val=0.01, dropout=0.1)
```

```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.1DO_0.01L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120
_________________________________________________________________
BatchNormalization1 (BatchNo (None, 32)                128
_________________________________________________________________
Dense2_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dropout_0.1 (Dropout)        (None, 32)                0
_________________________________________________________________
Dense3_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dense4_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
BatchNormalization2 (BatchNo (None, 32)                128
_________________________________________________________________
Dense5_LeakyReLU (Dense)     (None, 16)                528
_________________________________________________________________
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128
_________________________________________________________________
Total train time for 30 epochs = 74.720 seconds
```

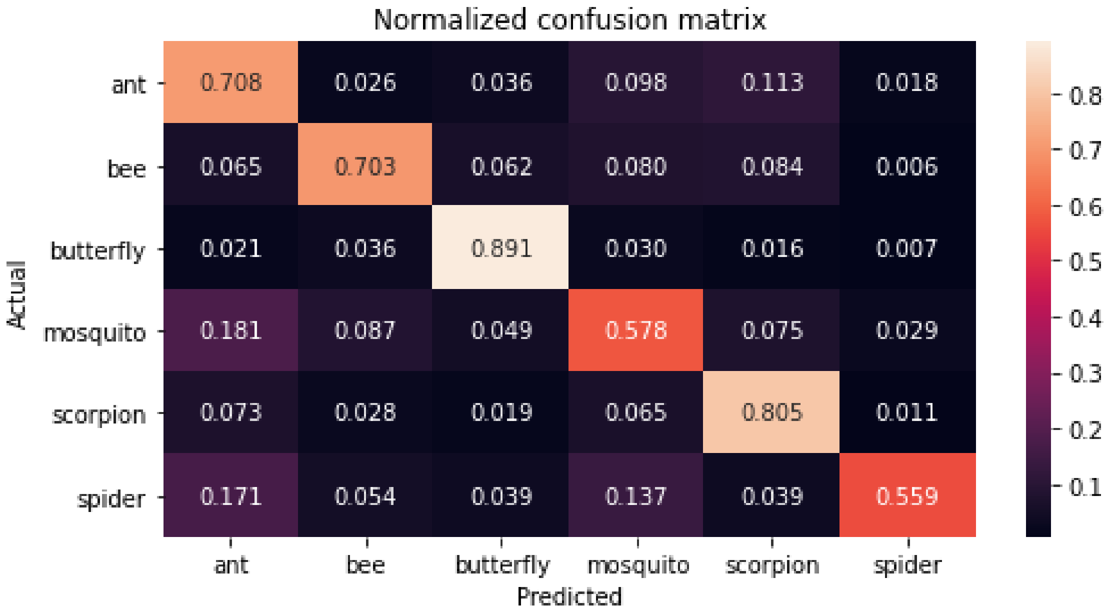


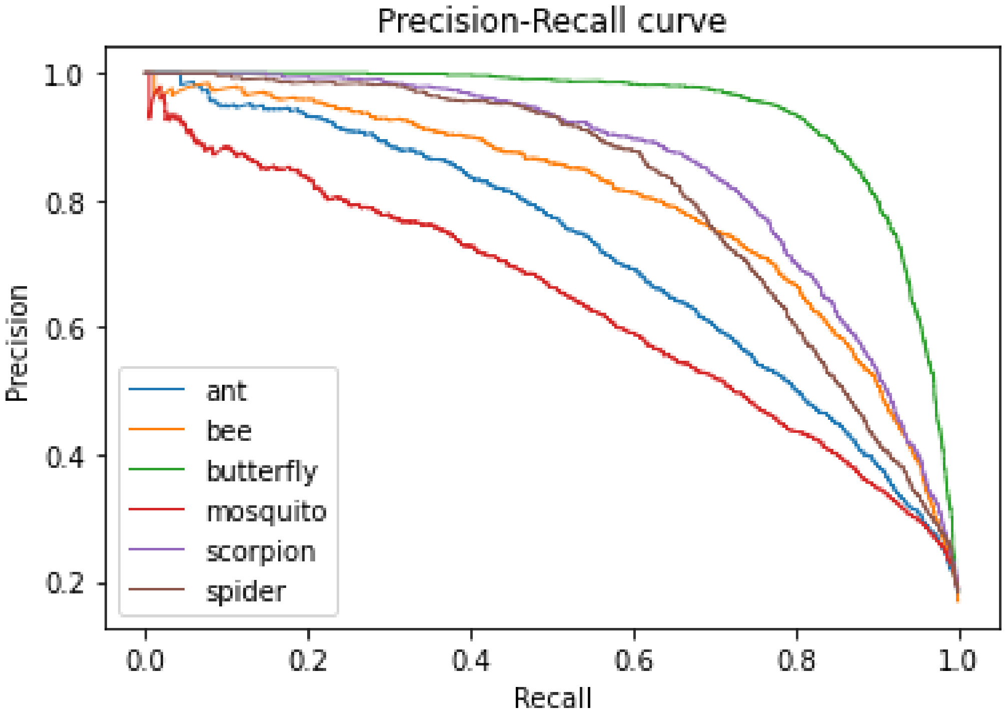
```
Test accuracy: 0.7078
```

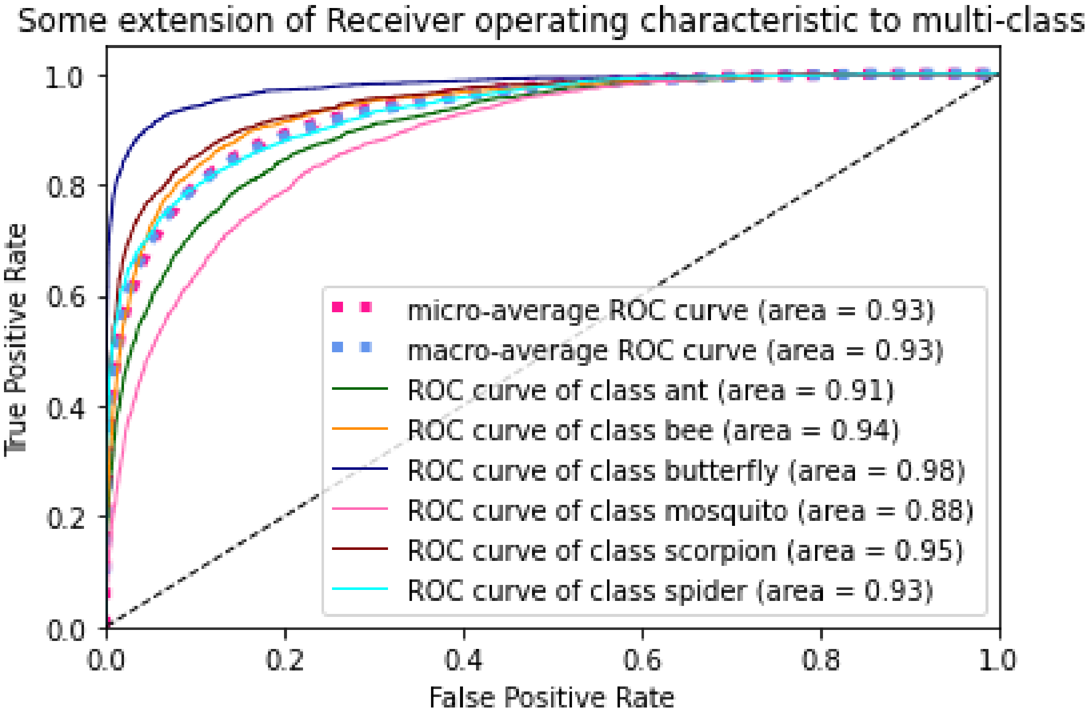


```
Test loss: 0.932
```

```
              precision    recall  f1-score   support

         ant       0.58      0.71      0.64      2478
         bee       0.75      0.70      0.73      2488
   butterfly       0.82      0.89      0.85      2557
    mosquito       0.59      0.58      0.58      2527
    scorpion       0.71      0.80      0.75      2468
      spider       0.89      0.56      0.69      2482

    accuracy                           0.71     15000
   macro avg       0.72      0.71      0.71     15000
weighted avg       0.72      0.71      0.71     15000
```


Normalized confusion matrix
Actual
ant
bee
butterfly
mosquito
scorpion
spider
0.708 0.026 0.036 0.098 0.113 0.018
0.065 0.703 0.062 0.080 0.084 0.006
0.021 0.036 0.891 0.030 0.016 0.007
0.181 0.087 0.049 0.578 0.075 0.029
0.073 0.028 0.019 0.065 0.805 0.011
0.171 0.054 0.039 0.137 0.039 0.559
ant
bee
butterfly
mosquito
scorpion
spider
Predicted
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

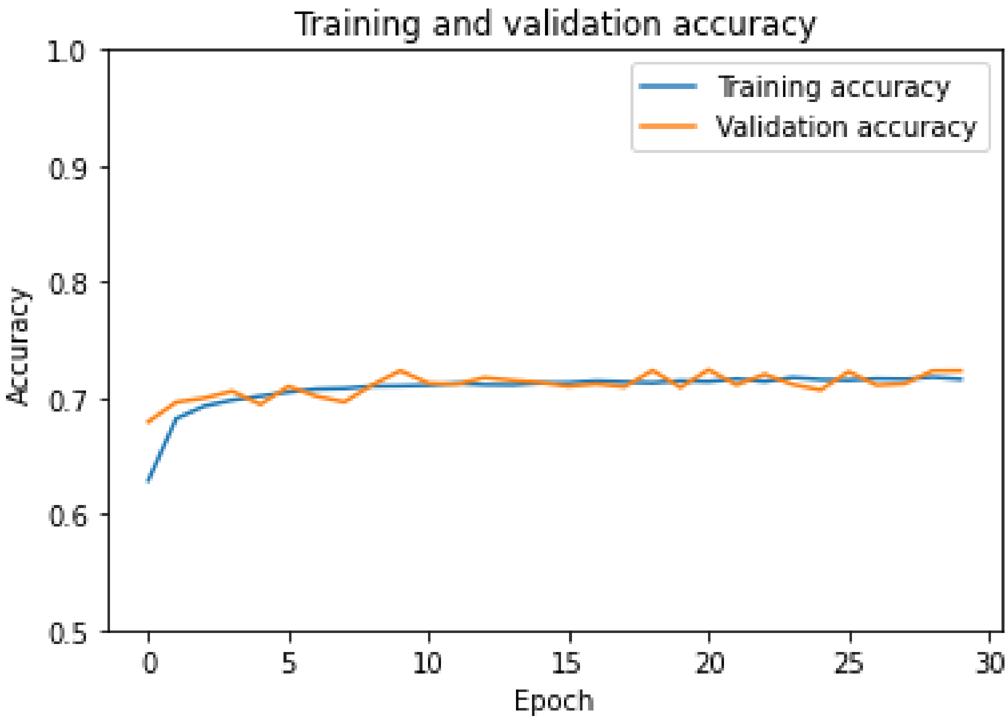

Precision-Recall curve
Precision
1.0
0.8
0.6
0.4
0.2
ant
bee
butterfly
mosquito
scorpion
spider
0.0
0.2
0.4
0.6
0.8
1.0
Recall



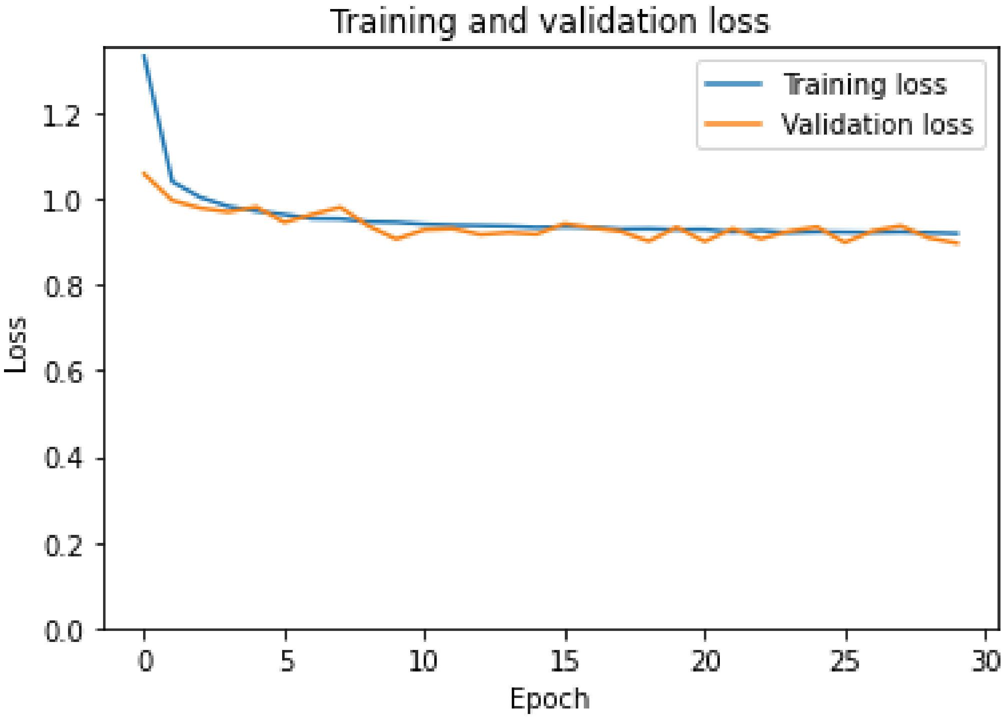
```
One-vs-One ROC AUC scores:
0.930512 (macro),
0.930593 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.930607 (macro),
0.930690 (weighted by prevalence)
```

```
[ ]: model_01L2_20do, history_01L2_20do, score_01L2_20do, y_pred_01L2_20do =␣
      ↪run_model(l2_val=0.01, dropout=0.2)
```

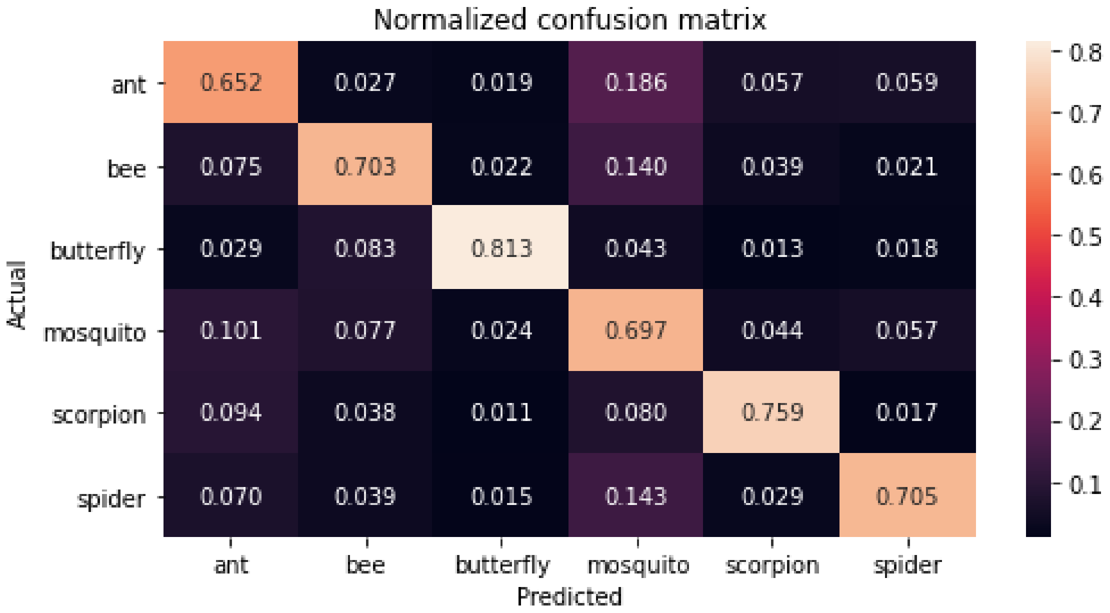
```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.2DO_0.01L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120
_________________________________________________________________
BatchNormalization1 (BatchNo (None, 32)                128
_________________________________________________________________
Dense2_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dropout_0.2 (Dropout)        (None, 32)                0
_________________________________________________________________
Dense3_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dense4_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
BatchNormalization2 (BatchNo (None, 32)                128
_________________________________________________________________
Dense5_LeakyReLU (Dense)     (None, 16)                528
_________________________________________________________________
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128
_________________________________________________________________
Total train time for 30 epochs = 74.767 seconds
```

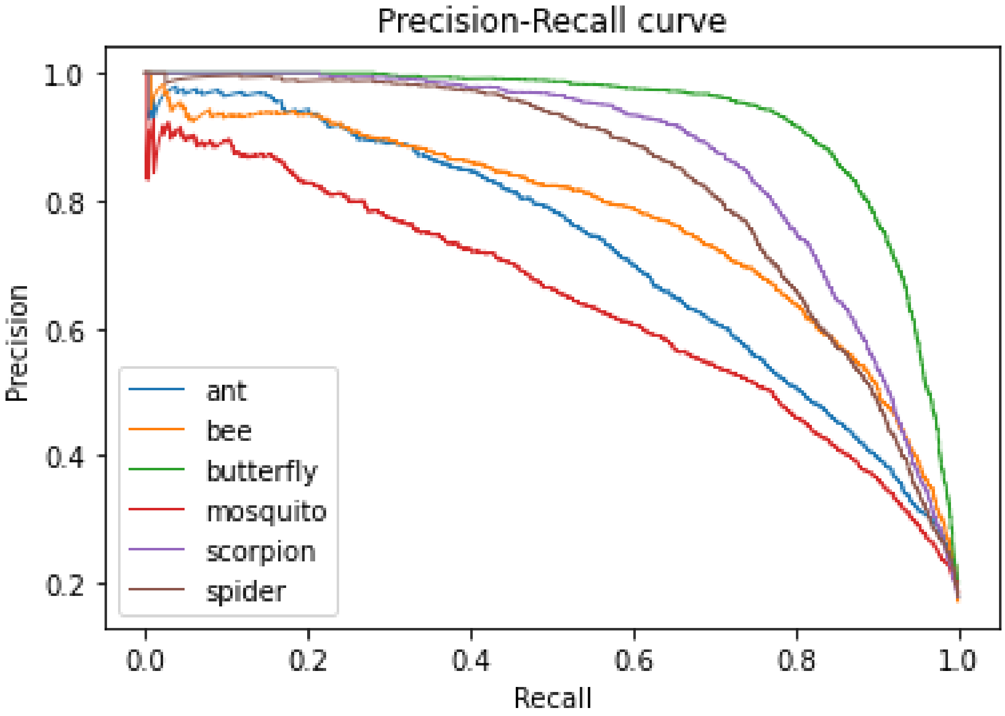


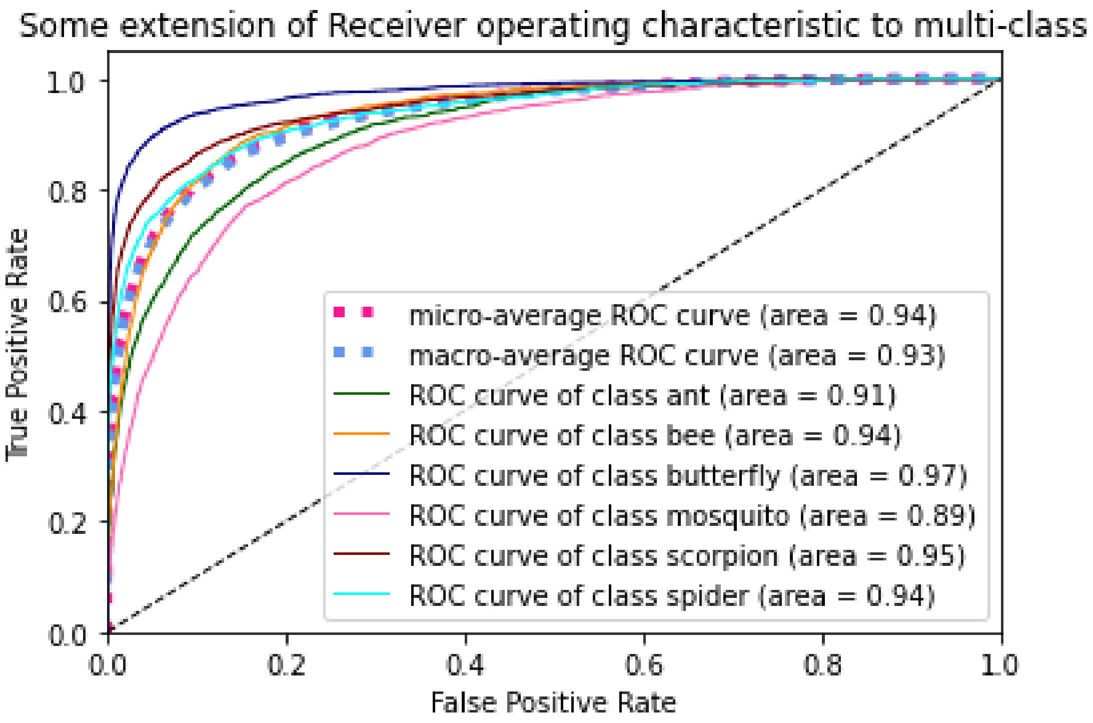
```
Test accuracy: 0.722
```



```
Test loss: 0.9059

              precision    recall  f1-score   support

         ant       0.64      0.65      0.64      2478
         bee       0.72      0.70      0.71      2488
   butterfly       0.90      0.81      0.86      2557
    mosquito       0.55      0.70      0.61      2527
    scorpion       0.81      0.76      0.78      2468
      spider       0.80      0.70      0.75      2482

    accuracy                           0.72     15000
   macro avg       0.74      0.72      0.73     15000
weighted avg       0.74      0.72      0.73     15000
```


Normalized confusion matrix
Actual
Predicted
ant
bee
butterfly
mosquito
scorpion
spider
ant 0.652 0.027 0.019 0.186 0.057 0.059
bee 0.075 0.703 0.022 0.140 0.039 0.021
butterfly 0.029 0.083 0.813 0.043 0.013 0.018
mosquito 0.101 0.077 0.024 0.697 0.044 0.057
scorpion 0.094 0.038 0.011 0.080 0.759 0.017
spider 0.070 0.039 0.015 0.143 0.029 0.705
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

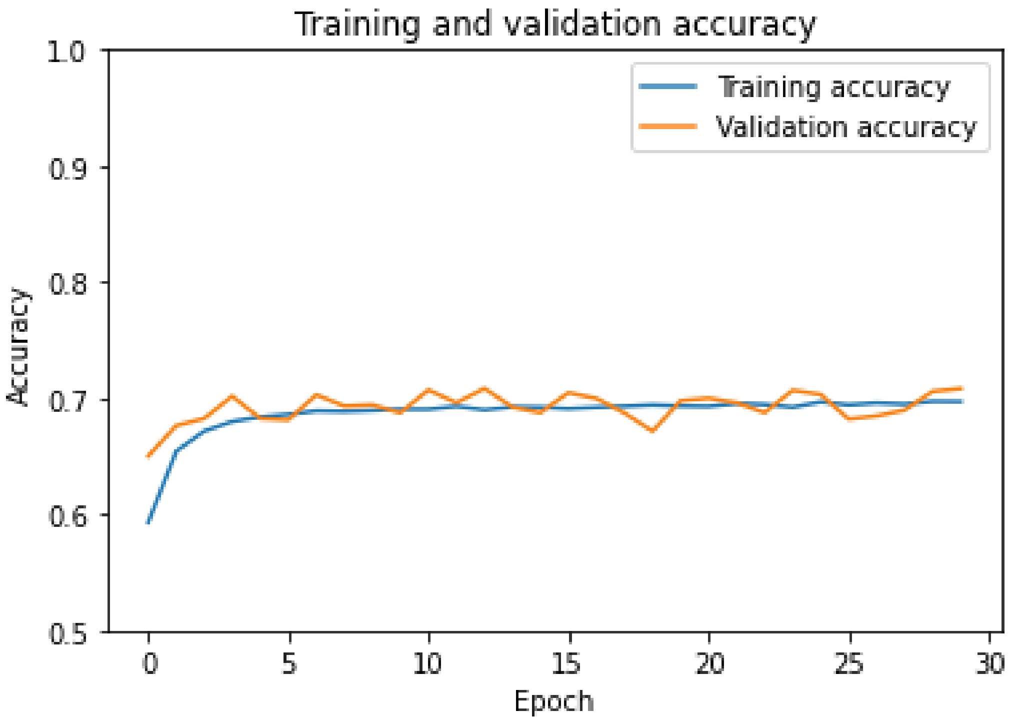

Precision-Recall curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
ant
bee
butterfly
mosquito
scorpion
spider



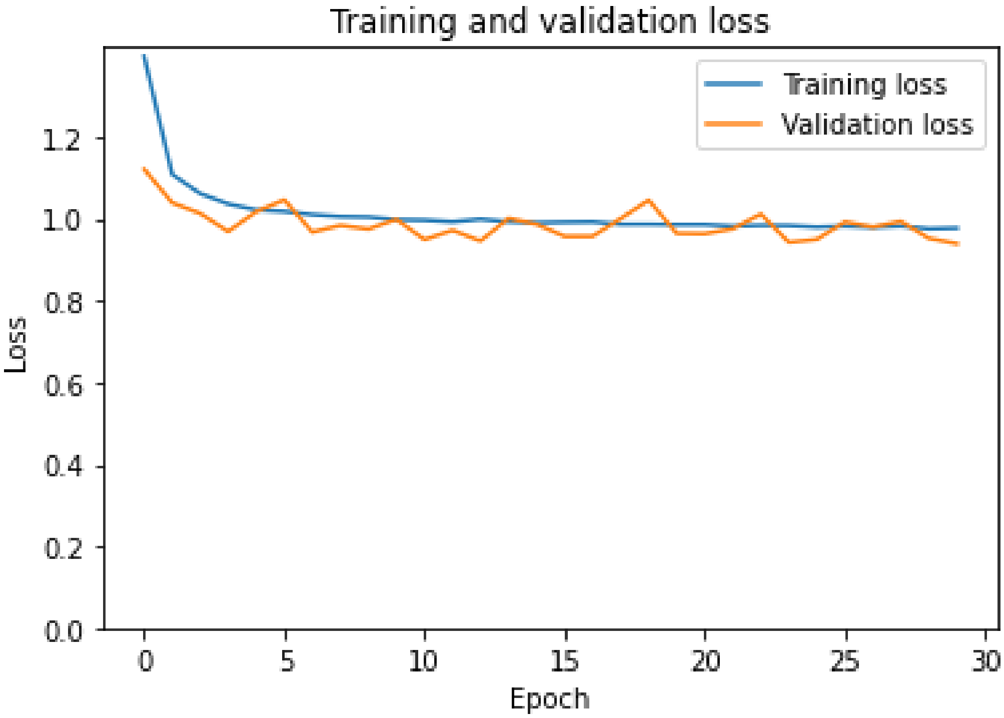
```
One-vs-One ROC AUC scores:
0.932608 (macro),
0.932666 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.932675 (macro),
0.932737 (weighted by prevalence)
```

[ ]:
```
model_01L2_40do, history_01L2_40do, score_01L2_40do, y_pred_01L2_40do =␣
↪run_model(l2_val=0.01, dropout=0.4)
```

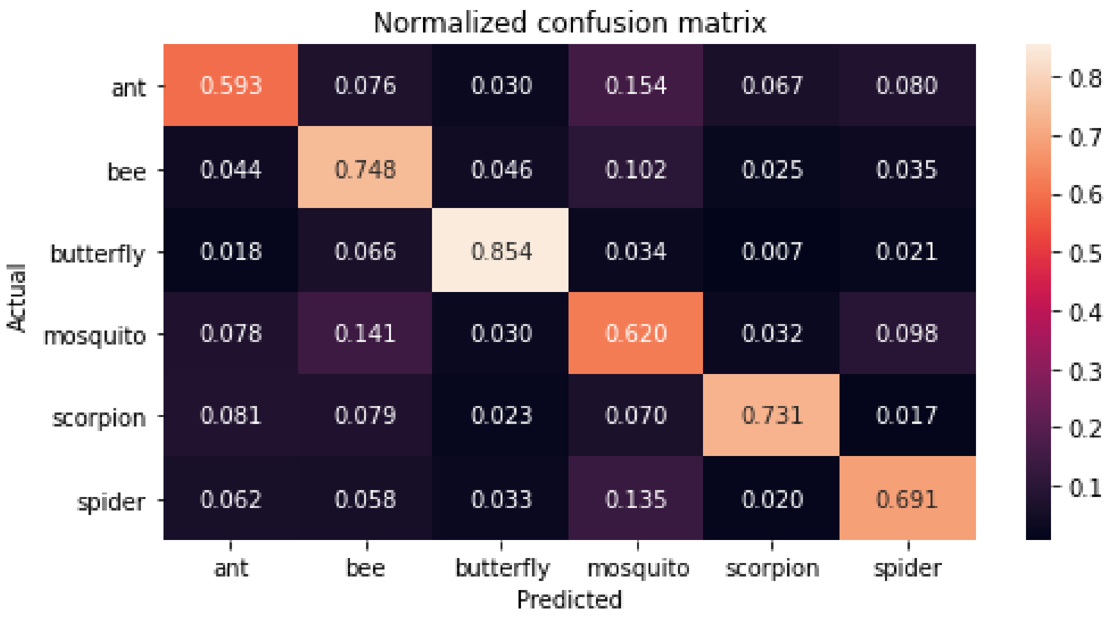
```
Model: "Multi_Layer_Perceptron_LeakyReLU_0.4DO_0.01L2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense1_LeakyReLU (Dense)     (None, 32)                25120
_________________________________________________________________
BatchNormalization1 (BatchNo (None, 32)                128
_________________________________________________________________
Dense2_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dropout_0.4 (Dropout)        (None, 32)                0
_________________________________________________________________
Dense3_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
Dense4_LeakyReLU (Dense)     (None, 32)                1056
_________________________________________________________________
BatchNormalization2 (BatchNo (None, 32)                128
_________________________________________________________________
Dense5_LeakyReLU (Dense)     (None, 16)                528
_________________________________________________________________
Output (Dense)               (None, 6)                 102
=================================================================
Total params: 29,174
Trainable params: 29,046
Non-trainable params: 128
_________________________________________________________________
Total train time for 30 epochs = 75.503 seconds
```

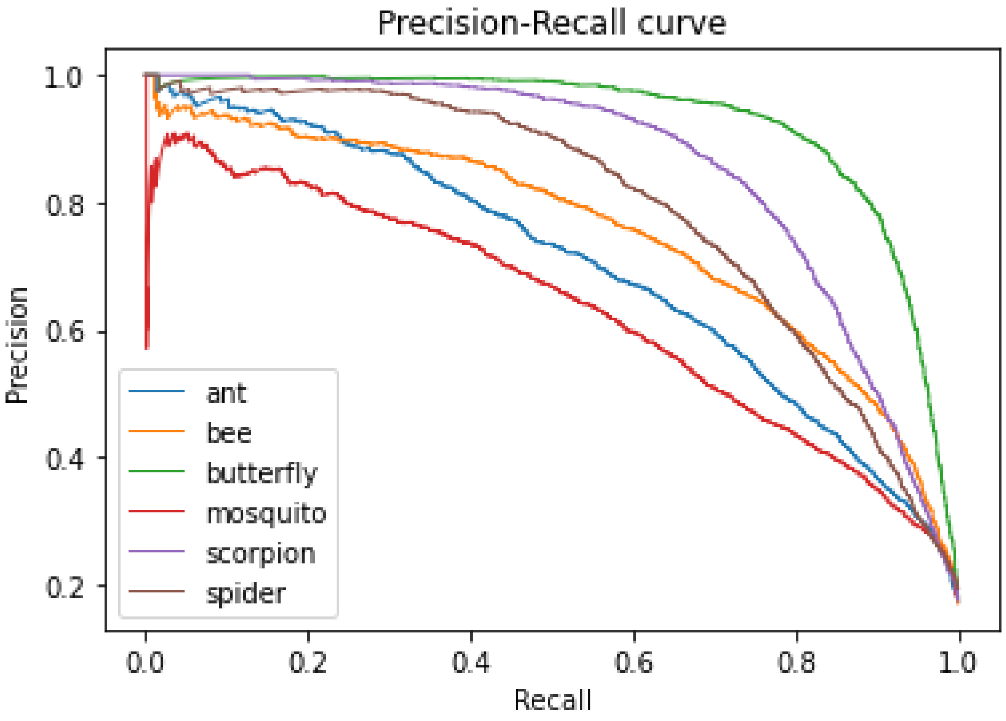


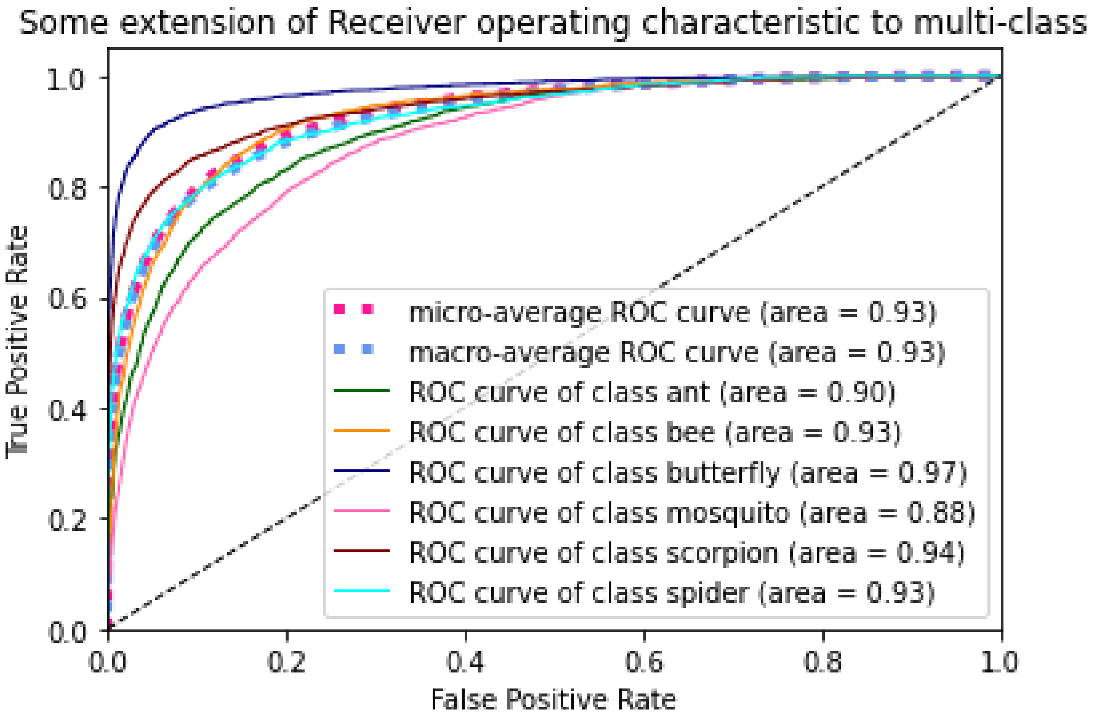
```
Test accuracy: 0.7068
```



```
Test loss: 0.9366

              precision    recall  f1-score   support

         ant       0.68      0.59      0.63      2478
         bee       0.64      0.75      0.69      2488
   butterfly       0.84      0.85      0.85      2557
    mosquito       0.56      0.62      0.59      2527
    scorpion       0.83      0.73      0.78      2468
      spider       0.73      0.69      0.71      2482

    accuracy                           0.71     15000
   macro avg       0.71      0.71      0.71     15000
weighted avg       0.71      0.71      0.71     15000
```


Normalized confusion matrix
Actual
Predicted
ant
bee
butterfly
mosquito
scorpion
spider
0.593 0.076 0.030 0.154 0.067 0.080
0.044 0.748 0.046 0.102 0.025 0.035
0.018 0.066 0.854 0.034 0.007 0.021
0.078 0.141 0.030 0.620 0.032 0.098
0.081 0.079 0.023 0.070 0.731 0.017
0.062 0.058 0.033 0.135 0.020 0.691
ant
bee
butterfly
mosquito
scorpion
spider
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Precision-Recall curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
ant
bee
butterfly
mosquito
scorpion
spider



```
One-vs-One ROC AUC scores:
0.925344 (macro),
0.925445 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.925465 (macro),
0.925565 (weighted by prevalence)
```

0.5.1 Time taken

[ ]:

```
total_time = (time.time() - start_total_time) / 60
print(f"Total time %.3f" % total_time, "minutes")
```

```
Total time 15.270 minutes
```