Pokemon_Final_Project_SGD

August 14, 2022

[132]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[133]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib.image as mpimg
import zipfile
import os
from pathlib import Path
import re
import tensorflow as tf
from tensorflow import keras
from keras import layers
from PIL import Image
import plotly.express as px
import glob
```

[134]:
```
data = pd.read_csv('/content/pokemon.csv')
```

[135]:
```
data = data.sort_values(by=['Name'], ascending=True).reset_index(drop=True)
```

[136]:
```
directory = r'/content/drive/MyDrive/images/images/'
for filename in os.listdir(directory):
    if filename.endswith('.jpg'):
        prefix = filename.split('.jpg')[0]
        os.rename(os.path.join(directory, filename), os.path.join(directory,␣
 ↪prefix+'.png'))
    else:
        continue
```

[137]:
```
train_dir = r'/content/drive/MyDrive/images/images/'
train_path = Path(train_dir)
```

[138]:
```
files = list(train_path.glob('*.png'))
name = [os.path.split(x)[1] for x in list(train_path.glob('*.png'))]

pokemon_image_df = pd.concat([pd.Series(name, name='Name'), pd.Series(files,␣
↪name='Filepath').astype(str)], axis=1)
pokemon_image_df['Name'] = pokemon_image_df['Name'].apply(lambda x: re.sub(r'\.
↪\w+$', '', x))
pokemon_image_df
```

[138]:

	Name	Filepath
0	aurorus	/content/drive/MyDrive/images/images/aurorus.png
1	arcanine	/content/drive/MyDrive/images/images/arcanine.png
2	abra	/content/drive/MyDrive/images/images/abra.png
3	barbaracle	/content/drive/MyDrive/images/images/barbaracl…
4	amoonguss	/content/drive/MyDrive/images/images/amoonguss…
..	…	…
804	vikavolt	/content/drive/MyDrive/images/images/vikavolt.png
805	wishiwashi-solo	/content/drive/MyDrive/images/images/wishiwash…
806	yungoos	/content/drive/MyDrive/images/images/yungoos.png
807	type-null	/content/drive/MyDrive/images/images/type-null…
808	zeraora	/content/drive/MyDrive/images/images/zeraora.png

[809 rows x 2 columns]

[139]:
```
#list(train_path.glob('*.png'))
comb_df = pokemon_image_df.merge(data, on='Name')
comb_df = comb_df.drop(['Name','Type2'], axis=1)
comb_df
```

[139]:

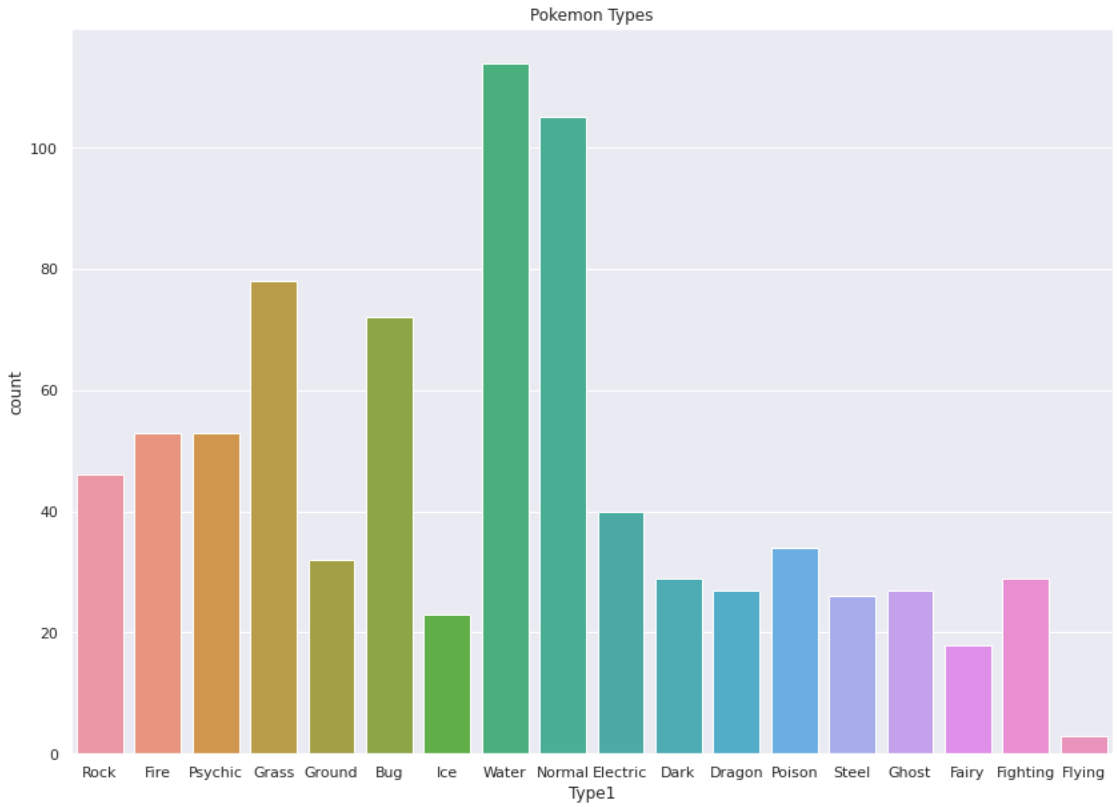
	Filepath	Type1
0	/content/drive/MyDrive/images/images/aurorus.png	Rock
1	/content/drive/MyDrive/images/images/arcanine.png	Fire
2	/content/drive/MyDrive/images/images/abra.png	Psychic
3	/content/drive/MyDrive/images/images/barbaracl…	Rock
4	/content/drive/MyDrive/images/images/amoonguss…	Grass
..	…	…
804	/content/drive/MyDrive/images/images/vikavolt.png	Bug
805	/content/drive/MyDrive/images/images/wishiwash…	Water
806	/content/drive/MyDrive/images/images/yungoos.png	Normal
807	/content/drive/MyDrive/images/images/type-null…	Normal
808	/content/drive/MyDrive/images/images/zeraora.png	Electric

[809 rows x 2 columns]

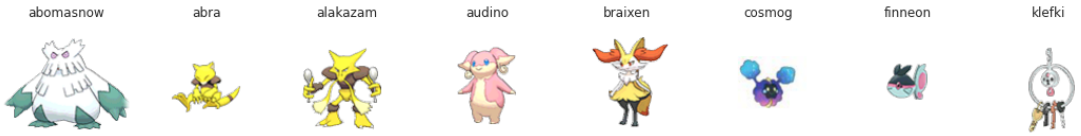
[140]:
```
data['Type1'].unique()
```

[140]:
```
array(['Grass', 'Psychic', 'Dark', 'Bug', 'Steel', 'Rock', 'Normal',
       'Water', 'Dragon', 'Electric', 'Poison', 'Fire', 'Fairy', 'Ice',
       'Ground', 'Ghost', 'Fighting', 'Flying'], dtype=object)
```

[141]:
```
sns.set(style="darkgrid")
sns.countplot(x="Type1", data=comb_df).set(title='Pokemon Types')
fig = plt.gcf()
# Change seaborn plot size
fig.set_size_inches(14, 10)
```



[142]:
```
path = r'/content/drive/MyDrive/images/images/'
fig,(ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8) = plt.subplots(1, 8, figsize=(15,␣
↪10))
ax = [ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8]
for i in range(8):
    img = mpimg.imread(path+data['Name'][i**3]+'.png', 0)
    ax[i].imshow(img)
    ax[i].set_title(data['Name'][i**3])
    ax[i].axis('off')
plt.tight_layout()
plt.show()
```



[143]:
```
type_colors = [
"#8ED752",
"#F95643",
"#53AFFE",
"#C3D221",
"#BBBDAF",
"#AD5CA2",
"#F8E64E",
"#F0CA42",
"#F9AEFE",
"#A35449",
"#FB61B4",
"#CDBD72",
"#7673DA",
"#66EBFF",
"#8B76FF",
"#8E6856",
"#C3C1D7",
"#75A4F9"]
```

[144]:
```
pokemon_types=data['Type1'].unique();␣
↪pokemon_colors=dict(zip(pokemon_types,type_colors))
```

[145]:
```
data['Type1'].value_counts()
```

[145]:
```
Water       114
Normal      105
Grass        78
Bug          72
Fire         53
Psychic      53
Rock         46
Electric     40
Poison       34
Ground       32
Dark         29
Fighting     29
Dragon       27
Ghost        27
```

```
Steel        26
Ice          23
Fairy        18
Flying        3
Name: Type1, dtype: int64
```

[146]:
```
# px.density_heatmap(comb_df, x="Type1", marginal_x="histogram", title='Pokemon Types')
```

[147]:
```
firewater = comb_df.query("Type1 == 'Fire' | Type1 == 'Water'")
firewater

print("Number of Fire Types:", len(firewater[firewater['Type1'] == 'Fire']))
print("Number of Water Types:", len(firewater[firewater['Type1'] == 'Water']))
```

```
Number of Fire Types: 53
Number of Water Types: 114
```

[148]:
```
# grasswater = comb_df.query("Type1 == 'Grass' | Type1 == 'Water'")
# grasswater

# print("Number of Fire Types:", len(grasswater[grasswater['Type1'] == 'Grass']))
# print("Number of Water Types:", len(grasswater[grasswater['Type1'] == 'Water']))
```

[149]:
```
# import os
# from shutil import copyfile
# os.mkdir('train/')
# os.mkdir('test/')
# os.mkdir('val/')
# for class_ in firewater['Type1'].unique():
#     os.mkdir('train/'+str(class_)+'/')
#     os.mkdir('test/'+str(class_)+'/')
#     os.mkdir('val/'+str(class_)+'/')
```

[150]:
```
# from sklearn.model_selection import train_test_split
# X_train, X_test, y_train, y_test = train_test_split(
#     firewater, firewater['Type1'],test_size=0.33, stratify=firewater['Type1'])

# X_test, X_val, y_test, y_val = train_test_split(
#     X_test, y_test, test_size=0.33,stratify=y_test)
```

[151]:
```
# from shutil import copyfile, copy2

# for image,type_  in zip(X_train['Filepath'], y_train):
#     copy2(image, 'train/'+type_)
```

```
# for image,type_ in zip(X_test['Filepath'], y_test):
#     copy2(image, 'test/'+type_)

# for image,type_ in zip(X_val['Filepath'], y_val):
#     copy2(image, 'val/'+type_)
```

[152]:
```
# from keras.preprocessing.image import ImageDataGenerator
# datagen = ImageDataGenerator(rescale=1./255)

# train = datagen.flow_from_directory('train/',
#     target_size=(120, 120),
#     color_mode='rgba',
#     class_mode='sparse',
#     shuffle=True,
#     seed=33)

# test = datagen.flow_from_directory('test/',
#     target_size=(120, 120),
#     color_mode='rgba',
#     class_mode='sparse',
#     shuffle=True,
#     seed=33)

# val = datagen.flow_from_directory('val/',
#     target_size=(120, 120),
#     color_mode='rgba',
#     class_mode='sparse',
#     shuffle=True,
#     seed=33)
```

[153]:
```
# from keras.models import Sequential
# from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense,␣
 ↪Activation, BatchNormalization, Lambda
# from keras.preprocessing.image import ImageDataGenerator

# def build():
#     model = Sequential()
#     IMAGE_WIDTH = 120
#     IMAGE_HEIGHT = 120
#     IMAGE_CHANNELS = 4
#     model.add(Lambda(lambda x: x, input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT,␣
 ↪IMAGE_CHANNELS)))
#     model.add(Conv2D(64, (4, 4), activation='relu'))
#     model.add(BatchNormalization())
#     model.add(MaxPooling2D(pool_size=(3, 3)))
#     model.add(Dropout(0.25))
```

```
#     model.add(Conv2D(128, (4, 4), activation='relu'))
#     model.add(BatchNormalization())
#     model.add(MaxPooling2D(pool_size=(3, 3)))
#     model.add(Dropout(0.25))

#     model.add(Conv2D(256, (4, 4), activation='relu'))
#     model.add(BatchNormalization())
#     model.add(MaxPooling2D(pool_size=(3, 3)))
#     model.add(Dropout(0.25))

#     model.add(Flatten())
#     model.add(Dense(512, activation='relu'))
#     model.add(BatchNormalization())
#     model.add(Dropout(0.5))
#     model.add(Dense(1, activation='softmax'))

# # If you have a multi-class classification problem = there is only one "right␣
 ↪answer" = the outputs are mutually exclusive, then use a softmax function.␣
 ↪The softmax will enforce that the sum of the probabilities of your output␣
 ↪classes are equal to one, so in order to increase the probability of a␣
 ↪particular class, your model must correspondingly decrease the probability␣
 ↪of at least one of the other classes. Example: classifying images from the␣
 ↪MNIST data set of handwritten digits. A single picture of a digit has only␣
 ↪one true identity - the picture cannot be a 7 and an 8 at the same time.
# # "Will there be any pictures that have BOTH cat and dog in the same picture?
 ↪" If the answer is yes (or unknown), use sigmoid. If the answer is no, use␣
 ↪softmax.

#     model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['acc', keras.metrics.AUC()])

# # https://keras.io/api/callbacks/early_stopping/

#     model.summary()
#     return model

# model = build()
# history = model.fit(train, epochs=30, validation_data=val,
#   callbacks=[keras.callbacks.EarlyStopping(monitor='val_loss',patience=3,
#       restore_best_weights=True), keras.callbacks.ReduceLROnPlateau()])
```

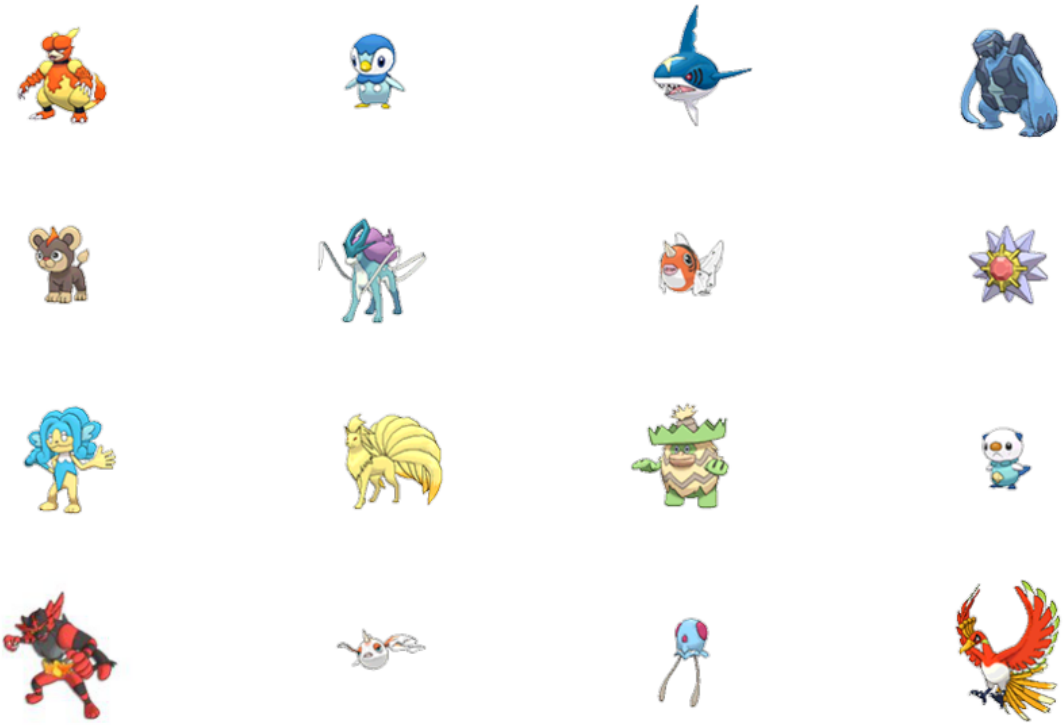
[154]:
```
#shuffle the data
firewater = firewater.sample(frac=1).reset_index(drop=True)

train_gen = keras.preprocessing.image.ImageDataGenerator(
    validation_split=0.2,   # split the dataset into a training set and a␣
 ↪validation set in an 8:2 ratio
```

```
    rescale=1./255          # rescale the rgb values to fit between 0 and 1
)
```

[155]:
```
train_data = train_gen.flow_from_dataframe(
    firewater,
    x_col='Filepath',
    y_col='Type1',
    target_size=(120, 120),
    color_mode='rgba',
    class_mode='sparse',
    batch_size=32,
    subset='training'
)

val_data = train_gen.flow_from_dataframe(
    firewater,
    x_col='Filepath',
    y_col='Type1',
    target_size=(120, 120),
    color_mode='rgba',
    class_mode='sparse',
    batch_size=32,
    subset='validation'
)
```

```
Found 134 validated image filenames belonging to 2 classes.
Found 33 validated image filenames belonging to 2 classes.
```

[156]:
```
sample_image = train_data.next()[0]

plt.figure(figsize=(16,10))
for i in range(16):
    plt.subplot(4, 4, i + 1)
    plt.imshow(sample_image[i, :, :, :])
    plt.axis('off')
plt.show()
```



[157]:
```
# inputs = tf.keras.Input(shape=(120, 120, 4))

# conv1 = tf.keras.layers.Conv2D(filters=64, kernel_size=(9,9),␣
 ↪activation='relu')(inputs)
# pool1 = tf.keras.layers.MaxPool2D()(conv1)

# conv2 = tf.keras.layers.Conv2D(filters=128, kernel_size=(9,9),␣
 ↪activation='relu')(pool1)
# pool2 = tf.keras.layers.MaxPool2D()(conv2)

# conv3 = tf.keras.layers.Conv2D(filters=256, kernel_size=(9,9),␣
 ↪activation='relu')(pool2)
# pool3 = tf.keras.layers.MaxPool2D()(conv3)

# outputs = tf.keras.layers.GlobalAveragePooling2D()(pool3)

# feature_extractor = tf.keras.Model(inputs=inputs, outputs=outputs)
```

[158]:
```
# feature_extractor.summary()
```

[159]:
```
inputs = layers.Input(shape=(120, 120, 4))

x = layers.Conv2D(filters=64, kernel_size=(9, 9), activation='relu')(inputs)
```

```
x = layers.MaxPool2D()(x)

x = layers.Conv2D(filters=128, kernel_size=(9, 9), activation='relu')(x)
x = layers.MaxPool2D()(x)

x = layers.Conv2D(filters=256, kernel_size=(9, 9), activation='relu')(x)
x = layers.MaxPool2D()(x)

x = layers.Flatten()(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.5)(x)

output = layers.Dense(units=1, activation='sigmoid')(x)

model = keras.Model(inputs=inputs, outputs=output)

model.compile(
    optimizer='SGD',
    loss='binary_crossentropy',
    metrics=['acc', keras.metrics.AUC()]
)

# print model layers
model.summary()
```

```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 120, 120, 4)]     0

 conv2d_15 (Conv2D)          (None, 112, 112, 64)      20800

 max_pooling2d_15 (MaxPoolin  (None, 56, 56, 64)       0
 g2D)

 conv2d_16 (Conv2D)          (None, 48, 48, 128)       663680

 max_pooling2d_16 (MaxPoolin  (None, 24, 24, 128)      0
 g2D)

 conv2d_17 (Conv2D)          (None, 16, 16, 256)       2654464

 max_pooling2d_17 (MaxPoolin  (None, 8, 8, 256)        0
 g2D)

 flatten_5 (Flatten)         (None, 16384)             0
```

```
 dense_10 (Dense)            (None, 512)               8389120

 dropout_5 (Dropout)         (None, 512)               0

 dense_11 (Dense)            (None, 1)                 513

=================================================================
Total params: 11,728,577
Trainable params: 11,728,577
Non-trainable params: 0
_________________________________________________________________
```

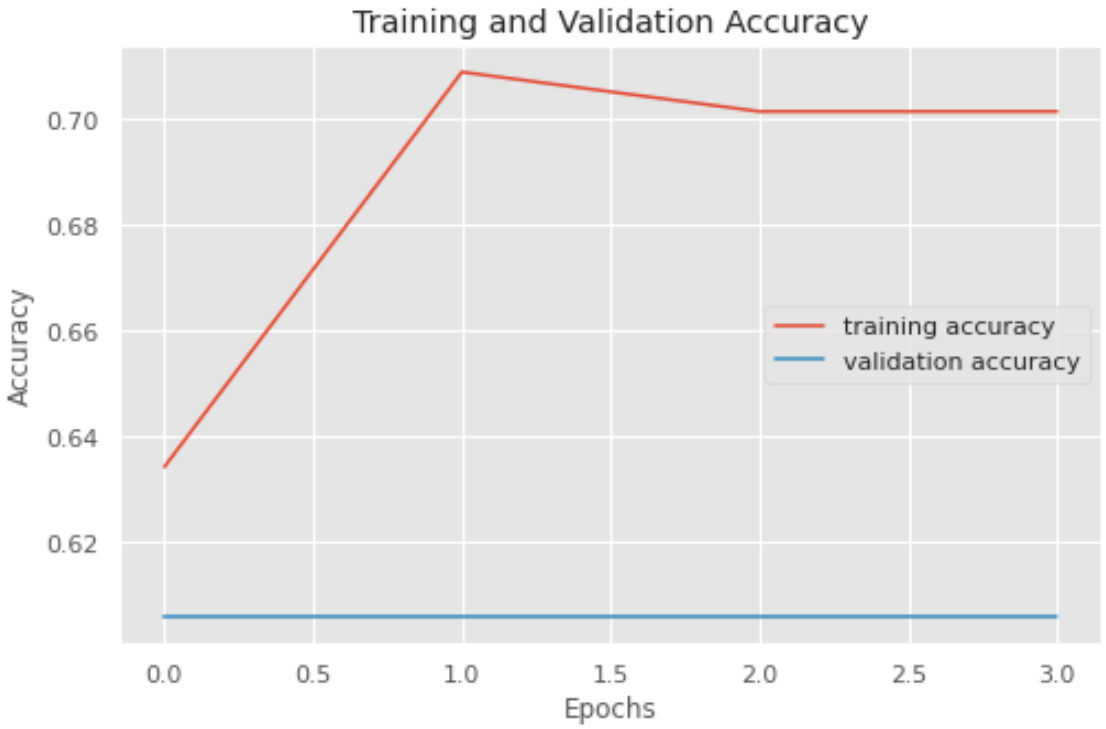
[160]:
```
history = model.fit(
    train_data,
    validation_data=val_data,
    epochs=30,
    callbacks=[
        keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=3,
            restore_best_weights=True
        ),
        keras.callbacks.ReduceLROnPlateau()
    ]
)
```

```
Epoch 1/30
5/5 [==============================] - 48s 9s/step - loss: 0.6349 - acc: 0.6343
- auc_5: 0.5424 - val_loss: 0.6819 - val_acc: 0.6061 - val_auc_5: 0.5538 - lr:
0.0100
Epoch 2/30
5/5 [==============================] - 46s 9s/step - loss: 0.5982 - acc: 0.7090
- auc_5: 0.6158 - val_loss: 0.8559 - val_acc: 0.6061 - val_auc_5: 0.7442 - lr:
0.0100
Epoch 3/30
5/5 [==============================] - 46s 11s/step - loss: 0.6220 - acc: 0.7015
- auc_5: 0.6342 - val_loss: 0.6968 - val_acc: 0.6061 - val_auc_5: 0.5750 - lr:
0.0100
Epoch 4/30
5/5 [==============================] - 47s 9s/step - loss: 0.5608 - acc: 0.7015
- auc_5: 0.7464 - val_loss: 0.7332 - val_acc: 0.6061 - val_auc_5: 0.6038 - lr:
0.0100
```

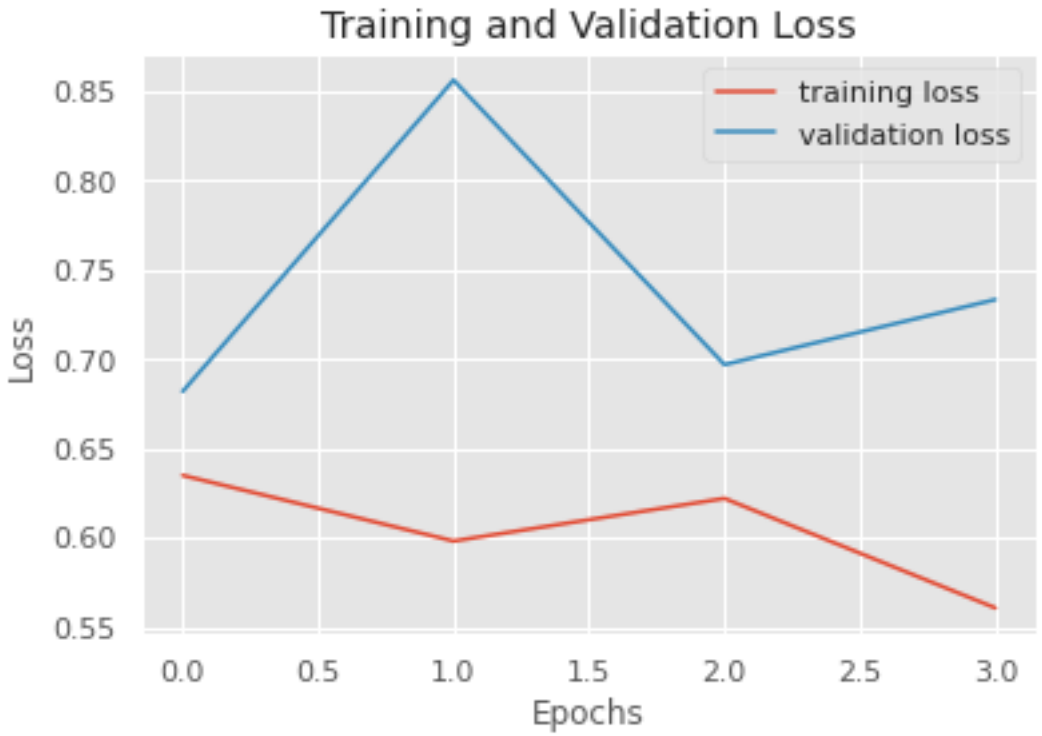
[161]:
```
plt.style.use('ggplot')

# retrieve accuracy history on training and validation data
acc = history.history['acc']
val_acc = history.history['val_acc']
```

```
# retrieve loss history on training and validation data
loss = history.history['loss']
val_loss = history.history['val_loss']

# get number of epochs
epochs = range(len(acc))

plt.figure(figsize=(8, 5))

# plot training and validation accuracy per epoch
plt.plot(epochs, acc, label='training accuracy')
plt.plot(epochs, val_acc, label='validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.figure(figsize=(8, 5))

# plot training and validation loss per epoch
plt.figure()
plt.plot(epochs, loss, label='training loss')
plt.plot(epochs, val_loss, label='validation loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

[161]: <matplotlib.legend.Legend at 0x7fc293950350>




```
<Figure size 576x360 with 0 Axes>
```



[162]:
```
# get true labels
true_labels = val_data.labels

# get predictions in the form of probablities
predictions = model.predict(val_data)

# convert probablities into binary values
predictions = [1 if n >= 0.5 else 0 for n in predictions]
print("Model predictions: "+str(predictions))
print("Actual labels:     "+str(true_labels))

# determine filepaths of misclassified pokemon
num_misclasssified = 0
misclassified_filepaths = []
correctness = []
for pred, label, i in zip(predictions, true_labels, range(len(predictions))):
  misclassified_filepaths.append(val_data.filepaths[i])
  if pred != label:
    correctness.append('incorrect')
    num_misclasssified += 1
  else:
    correctness.append('correct')

print("# of misclassified pokemon: "+str(num_misclasssified))
```

```
Model predictions: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Actual labels:     [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0,
0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0]
# of misclassified pokemon: 13
```

[163]:
```
# obtain the images from the filepath at the determined indices
misclassified_imgs = []
for filepath in misclassified_filepaths:
  misclassified_imgs.append(mpimg.imread(filepath, 0))

# plot results
f, axarr = plt.subplots(6,5, figsize=(20,10))
count = 0
for r in range(6):
  for c in range(5):
    axarr[r,c].imshow(misclassified_imgs[count])
    if correctness[count] == 'correct':
      axarr[r,c].set_title(correctness[count])
    else:
```

```
      axarr[r,c].set_title(correctness[count], color='red')
    axarr[r,c].set_axis_off()
    count += 1
plt.show()
```