In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
from sklearn.preprocessing import minmax_scale

import os
from pathlib import Path
import re

import tensorflow as tf
```

In [40]:
```
train_dir = '/content/pokemon'
train_path = Path(train_dir)
train_path
```

Out[40]:
```
PosixPath('/content/pokemon')
```

In [55]:
```
files = list(train_path.glob('*.png'))
names = [os.path.split(x)[1] for x in list(train_path.glob('*.png'))]

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

Out[55]:

	Name	Filepath
0	buneary	/content/pokemon/buneary.png
1	skiddo	/content/pokemon/skiddo.png
2	magikarp	/content/pokemon/magikarp.png
3	larvitar	/content/pokemon/larvitar.png
4	archeops	/content/pokemon/archeops.png
...	...	...
716	vulpix	/content/pokemon/vulpix.png
717	dragonair	/content/pokemon/dragonair.png
718	tauros	/content/pokemon/tauros.png
719	silcoon	/content/pokemon/silcoon.png
720	exploud	/content/pokemon/exploud.png

721 rows × 2 columns

In [56]:
```
pokemon_df = pd.read_csv('/content/pokemon.csv')
pokemon_df
```

Out[56]:

	Name	Type1	Type2
0	bulbasaur	Grass	Poison
1	ivysaur	Grass	Poison
2	venusaur	Grass	Poison
3	charmander	Fire	NaN
4	charmeleon	Fire	NaN
...	...	...	...
804	stakataka	Rock	Steel
805	blacephalon	Fire	Ghost
806	zeraora	Electric	NaN
807	meltan	Steel	NaN
808	melmetal	Steel	NaN

809 rows × 3 columns

In [57]:
```
# Merging dfs
train_df = image_df.merge(pokemon_df, on='Name')
train_df = train_df.drop(['Name', 'Type2'], axis=1)
train_df
```

Out[57]:

	Filepath	Type1
0	/content/pokemon/buneary.png	Normal
1	/content/pokemon/skiddo.png	Grass
2	/content/pokemon/magikarp.png	Water
3	/content/pokemon/larvitar.png	Rock
4	/content/pokemon/archeops.png	Rock
...	...	...
716	/content/pokemon/vulpix.png	Fire
717	/content/pokemon/dragonair.png	Dragon
718	/content/pokemon/tauros.png	Normal
719	/content/pokemon/silcoon.png	Bug
720	/content/pokemon/exploud.png	Normal

721 rows × 2 columns

In [58]:
```
# Limiting data to Fire and Water types
train_df = train_df.query("Type1 == 'Fire' | Type1 == 'Water'")
train_df
```

Out[58]:

	Filepath	Type1
2	/content/pokemon/magikarp.png	Water
7	/content/pokemon/pansear.png	Fire
9	/content/pokemon/slugma.png	Fire
11	/content/pokemon/squirtle.png	Water
12	/content/pokemon/poliwrath.png	Water
...	...	...
687	/content/pokemon/tentacool.png	Water
701	/content/pokemon/wailmer.png	Water
702	/content/pokemon/poliwhirl.png	Water
704	/content/pokemon/palkia.png	Water
716	/content/pokemon/vulpix.png	Fire

152 rows × 2 columns

In [59]:
```
train_df.Type1.value_counts()
```

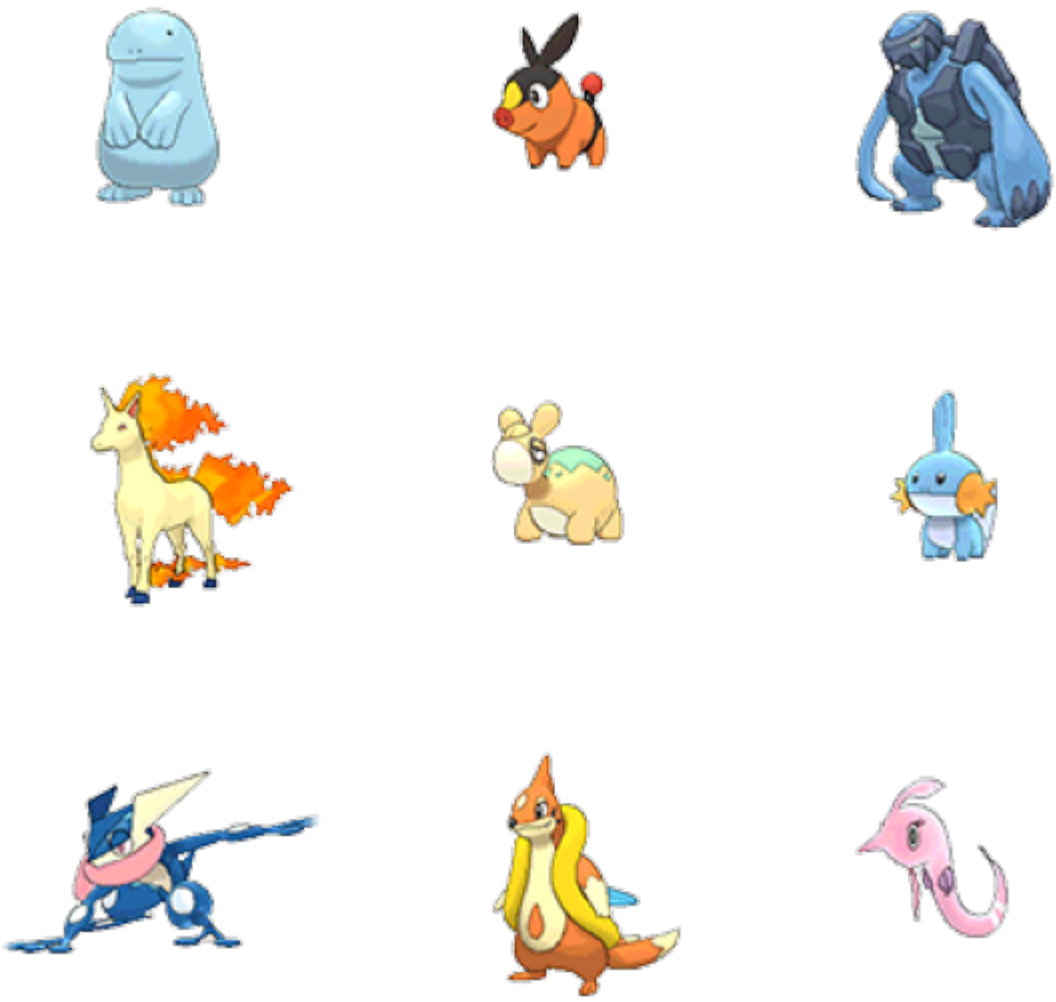
Out[59]:
```
Water    105
Fire      47
Name: Type1, dtype: int64
```

In [60]:
```
train_gen = tf.keras.preprocessing.image.ImageDataGenerator(
    validation_split=0.2,
    rescale=1./255
)
```

In [61]:
```
train_data = train_gen.flow_from_dataframe(
    train_df,
    x_col='Filepath',
    y_col='Type1',
    target_size=(120, 120),
    color_mode='rgba',
    class_mode='sparse',
    batch_size=32,
    shuffle=True,
    seed=1,
    subset='training'
)

val_data = train_gen.flow_from_dataframe(
    train_df,
    x_col='Filepath',
    y_col='Type1',
    target_size=(120, 120),
    color_mode='rgba',
    class_mode='sparse',
    batch_size=32,
    shuffle=True,
    seed=1,
    subset='validation'
)
```

```
Found 122 validated image filenames belonging to 2 classes.
Found 30 validated image filenames belonging to 2 classes.
```

In [62]:
```
image_sample = train_data.next()[0]

plt.figure(figsize=(10, 10))
for i in range(9):
    plt.subplot(3, 3, i + 1)
    plt.imshow(image_sample[i, :, :, :])
    plt.axis('off')
plt.show()
```



In [63]:
```
inputs = tf.keras.Input(shape=(120, 120, 4))

conv1 = tf.keras.layers.Conv2D(filters=64, kernel_size=(8, 8), activation='relu')(inputs)
pool1 = tf.keras.layers.MaxPool2D()(conv1)

conv2 = tf.keras.layers.Conv2D(filters=128, kernel_size=(8, 8), activation='relu')(pool1)
pool2 = tf.keras.layers.MaxPool2D()(conv2)

conv3 = tf.keras.layers.Conv2D(filters=256, kernel_size=(8, 8), activation='relu')(pool2)
pool3 = tf.keras.layers.MaxPool2D()(conv3)

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


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

In [64]:
```
feature_extractor.summary()
```

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 120, 120, 4)]     0

 conv2d_6 (Conv2D)           (None, 113, 113, 64)      16448

 max_pooling2d_6 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 conv2d_7 (Conv2D)           (None, 49, 49, 128)       524416

 max_pooling2d_7 (MaxPooling  (None, 24, 24, 128)      0
 2D)

 conv2d_8 (Conv2D)           (None, 17, 17, 256)       2097408

 max_pooling2d_8 (MaxPooling  (None, 8, 8, 256)        0
 2D)

 global_average_pooling2d_2   (None, 256)              0
 (GlobalAveragePooling2D)

=================================================================
Total params: 2,638,272
Trainable params: 2,638,272
Non-trainable params: 0
_________________________________________________________________
```

In [65]:
```
clf_inputs = feature_extractor.input
clf_outputs = tf.keras.layers.Dense(units=1, activation='sigmoid')(feature_ext
ractor.output)

classifier = tf.keras.Model(inputs=clf_inputs, outputs=clf_outputs)
```

In [66]:
```
classifier.summary()
```

```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 120, 120, 4)]     0

 conv2d_6 (Conv2D)           (None, 113, 113, 64)      16448

 max_pooling2d_6 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 conv2d_7 (Conv2D)           (None, 49, 49, 128)       524416

 max_pooling2d_7 (MaxPooling  (None, 24, 24, 128)      0
 2D)

 conv2d_8 (Conv2D)           (None, 17, 17, 256)       2097408

 max_pooling2d_8 (MaxPooling  (None, 8, 8, 256)        0
 2D)

 global_average_pooling2d_2   (None, 256)              0
 (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 1)                 257

=================================================================
Total params: 2,638,529
Trainable params: 2,638,529
Non-trainable params: 0
_________________________________________________________________
```

In [68]:
```
classifier.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

history = classifier.fit(
    train_data,
    validation_data=val_data,
    batch_size=32,
    epochs=25,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=3,
            restore_best_weights=True
        ),
        tf.keras.callbacks.ReduceLROnPlateau()
    ]
)
```

```
Epoch 1/25
4/4 [==============================] - 38s 10s/step - loss: 0.5032 - accurac
y: 0.6885 - val_loss: 0.4124 - val_accuracy: 0.7000 - lr: 0.0010
Epoch 2/25
4/4 [==============================] - 38s 10s/step - loss: 0.4249 - accurac
y: 0.6885 - val_loss: 0.3431 - val_accuracy: 0.7000 - lr: 0.0010
Epoch 3/25
4/4 [==============================] - 37s 9s/step - loss: 0.3757 - accuracy:
0.6885 - val_loss: 0.3151 - val_accuracy: 0.7000 - lr: 0.0010
Epoch 4/25
4/4 [==============================] - 38s 10s/step - loss: 0.4046 - accurac
y: 0.7049 - val_loss: 0.3945 - val_accuracy: 0.8667 - lr: 0.0010
Epoch 5/25
4/4 [==============================] - 38s 9s/step - loss: 0.4597 - accuracy:
0.8033 - val_loss: 0.3217 - val_accuracy: 0.7667 - lr: 0.0010
Epoch 6/25
4/4 [==============================] - 38s 10s/step - loss: 0.3755 - accurac
y: 0.7295 - val_loss: 0.3123 - val_accuracy: 0.7333 - lr: 0.0010
Epoch 7/25
4/4 [==============================] - 38s 9s/step - loss: 0.3350 - accuracy:
0.7787 - val_loss: 0.2995 - val_accuracy: 0.8333 - lr: 0.0010
Epoch 8/25
4/4 [==============================] - 38s 10s/step - loss: 0.3173 - accurac
y: 0.8197 - val_loss: 0.2852 - val_accuracy: 0.8667 - lr: 0.0010
Epoch 9/25
4/4 [==============================] - 37s 10s/step - loss: 0.3104 - accurac
y: 0.8525 - val_loss: 0.2694 - val_accuracy: 0.8667 - lr: 0.0010
Epoch 10/25
4/4 [==============================] - 38s 10s/step - loss: 0.2941 - accurac
y: 0.8852 - val_loss: 0.2547 - val_accuracy: 0.8667 - lr: 0.0010
Epoch 11/25
4/4 [==============================] - 38s 10s/step - loss: 0.2773 - accurac
y: 0.8852 - val_loss: 0.2369 - val_accuracy: 0.8667 - lr: 0.0010
Epoch 12/25
4/4 [==============================] - 39s 10s/step - loss: 0.2746 - accurac
y: 0.8852 - val_loss: 0.2191 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 13/25
4/4 [==============================] - 37s 9s/step - loss: 0.2541 - accuracy:
0.8852 - val_loss: 0.2020 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 14/25
4/4 [==============================] - 39s 10s/step - loss: 0.2245 - accurac
y: 0.9016 - val_loss: 0.1918 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 15/25
4/4 [==============================] - 38s 10s/step - loss: 0.2171 - accurac
y: 0.9016 - val_loss: 0.1679 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 16/25
4/4 [==============================] - 40s 11s/step - loss: 0.2160 - accurac
y: 0.9016 - val_loss: 0.1648 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 17/25
4/4 [==============================] - 39s 9s/step - loss: 0.2203 - accuracy:
0.9098 - val_loss: 0.1857 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 18/25
4/4 [==============================] - 38s 10s/step - loss: 0.1959 - accurac
y: 0.9180 - val_loss: 0.1705 - val_accuracy: 0.9333 - lr: 0.0010
Epoch 19/25
4/4 [==============================] - 39s 10s/step - loss: 0.2000 - accurac
y: 0.9098 - val_loss: 0.1825 - val_accuracy: 0.9667 - lr: 0.0010
```

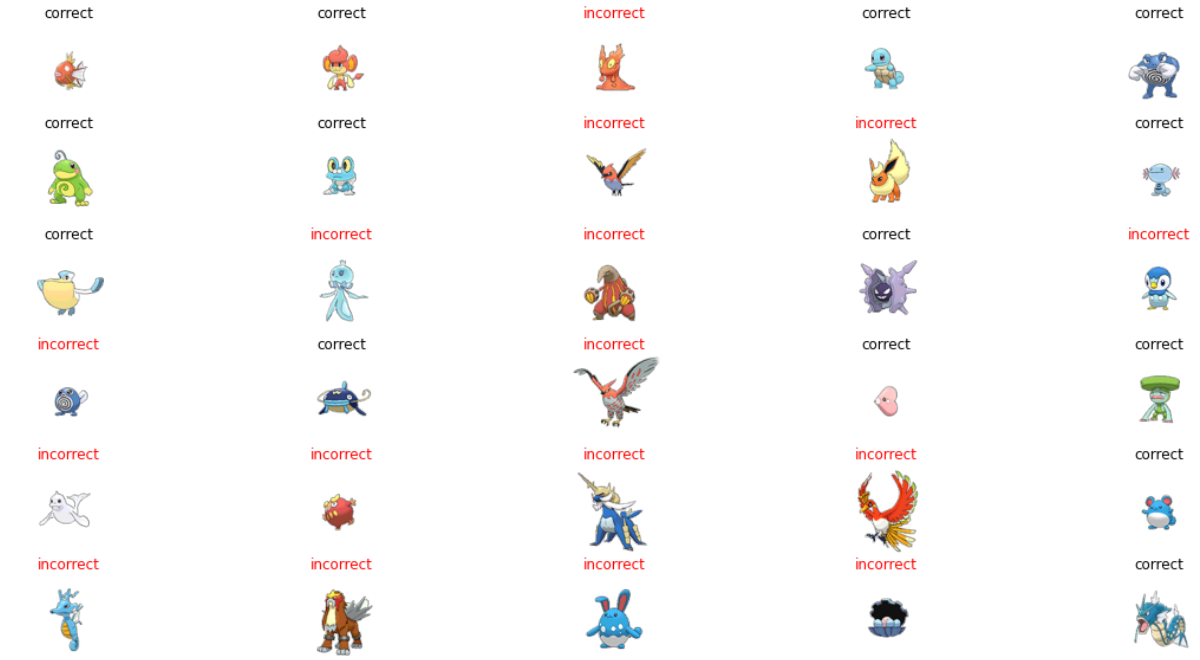
In [69]:
```
# retrieve accuracy history on training and validation data
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

# 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))

# 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.legend()

# 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.legend()
```

Out[69]: <matplotlib.legend.Legend at 0x7f533cf2d150>

In [70]:
```
# get true labels
true_labels = val_data.labels

# get predictions in the form of probablities
predictions = classifier.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, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1]
Actual labels:     [1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1]
# of misclassified pokemon: 16
```

In [72]:
```
# obtain the images from the filepath at the determined indices
misclassified_imgs = []
for filepath in misclassified_filepaths:
  misclassified_imgs.append(mpimg.imread(filepath))

# 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()
```



In [ ]: