원데이터를 데이터 증강을 이용하여 각 데이터를 6000개씩 증강 총 약 60000개의 사진으로 데이터 학습 및 검증

['Charmander:파이리', 'Diglett:디그다', 'Ditto:메타몽', 'Eevee:이브이', 'Gyarados:갸라도스', 'Meowth:나옹', 'Pikachu:피카츄', 'Rattata:꼬렛', 'Snorlax:잠만보', 'Squirtle:꼬부기'] 데이터 사용

In [306]:
```
import warnings
warnings.filterwarnings('ignore')

from keras import models, layers
import cv2
from glob import glob
import os
import numpy as np
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
import tensorflow as tf

from keras import regularizers
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.callbacks import ModelCheckpoint,EarlyStopping
import matplotlib.pyplot as plt

import Augmentor
import random
from PIL import Image
import PIL.ImageOps
import time
```

Charmander 데이터셋 6000개 늘리기 (파이리)

In [307]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Charmander₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-307-eb4c4cf93c93> in <module>
      1 num_augmented_images = 6000
      2 file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Charmander₩₩'
----> 3 file_names = os.listdir(file_path)
      4 total_origin_image_num = len(file_names)
      5 augment_cnt = 1

FileNotFoundError: [WinError 3] 지정된 경로를 찾을 수 없습니다: 'D:₩₩swproject₩₩pocketmon_classi₩₩Charmander₩₩'
```

In [5]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Charmander₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
197
Charmander.277.jpg

D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.277.jpg
rotate
109
Charmander.198.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.198.jpg
invert
148
Charmander.232.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.232.jpg
noise
259
Charmander.66.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.66.jpg
noise
116
Charmander.203.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.203.jpg
```

Digrett 데이터셋 6000개 늘리기 (디그다)

In [8]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Diglett₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [9]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Diglett₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
Diglett.42.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.42.jpg
invert
1
Diglett.10.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.10.jpg
invert
47
Diglett.6.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.6.jpg
noise
31
Diglett.38.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.38.jpg
rotate
48
Diglett.7.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.7.jpg
invert
42
```

Ditto 데이터셋 6000개 늘리기 (메타몽)

In [10]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Ditto₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [11]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Ditto₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
18
Ditto.26.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.26.jpg
invert
32
Ditto.39.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.39.jpg
noise
32
Ditto.39.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.39.jpg
rotate
41
Ditto.47.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.47.jpg
rotate
8
Ditto.17.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.17.jpg
```

Eevee 데이터셋 6000개 늘리기 (이브이)

In [12]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Eevee₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [13]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Eevee₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.34.jpg
rotate
35
Eevee.41.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.41.jpg
rotate
31
Eevee.38.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.38.jpg
noise
35
Eevee.41.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.41.jpg
rotate
31
Eevee.38.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.38.jpg
rotate
6
Eevee.15.jpg
```

Gyarados 데이터셋 6000개 늘리기 (갸라도스)

In [15]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Gyarados₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [16]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Gyarados₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
16
Gyarados.24.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.24.jpg
invert
21
Gyarados.29.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.29.jpg
invert
22
Gyarados.3.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.3.jpg
noise
6
Gyarados.15.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.15.jpg
rotate
47
Gyarados.52.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.52.jpg
noise
```

Meowth 데이터셋 6000개 늘리기 (나옹)

In [17]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩₩Meowth₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [18]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Meowth₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
50
Meowth.55.jpg
D:₩swproject₩pocketmon_classi₩Meowth₩Meowth.55.jpg
rotate
26
Meowth.33.jpg
D:₩swproject₩pocketmon_classi₩Meowth₩Meowth.33.jpg
invert
68
Meowth.8.jpg
D:₩swproject₩pocketmon_classi₩Meowth₩Meowth.8.jpg
invert
36
Meowth.42.jpg
D:₩swproject₩pocketmon_classi₩Meowth₩Meowth.42.jpg
noise
37
Meowth.43.jpg
D:₩swproject₩pocketmon_classi₩Meowth₩Meowth.43.jpg
```

Pikachu 데이터셋 6000개 늘리기 (피카츄)

In [19]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Pikachu₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [20]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Pikachu₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
272
Pikachu.76.jpg
D:₩swproject₩pocketmon_classi₩Pikachu₩Pikachu.76.jpg
noise
243
Pikachu.5.jpg
D:₩swproject₩pocketmon_classi₩Pikachu₩Pikachu.5.jpg
rotate
23
Pikachu.12.jpg
D:₩swproject₩pocketmon_classi₩Pikachu₩Pikachu.12.jpg
noise
150
Pikachu.234.jpg
D:₩swproject₩pocketmon_classi₩Pikachu₩Pikachu.234.jpg
noise
160
Pikachu.243.jpg
D:₩swproject₩pocketmon_classi₩Pikachu₩Pikachu.243.jpg
```

Rattata 데이터셋 6000개 늘리기 (꼬렛)

In [21]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Rattata₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [22]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Rattata₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
Rattata.8.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.8.jpg
rotate
28
Rattata.35.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.35.jpg
rotate
4
Rattata.13.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.13.jpg
invert
14
Rattata.22.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.22.jpg
noise
49
Rattata.54.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.54.jpg
rotate
```

Snorlax 데이터셋 6000개 늘리기 (잠만보)

In [23]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Snorlax₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [24]:
```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Snorlax₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
64
Snorlax.68.jpg
D:₩swproject₩pocketmon_classi₩Snorlax₩Snorlax.68.jpg
invert
59
Snorlax.63.jpg
D:₩swproject₩pocketmon_classi₩Snorlax₩Snorlax.63.jpg
noise
28
Snorlax.35.jpg
D:₩swproject₩pocketmon_classi₩Snorlax₩Snorlax.35.jpg
noise
30
Snorlax.37.jpg
D:₩swproject₩pocketmon_classi₩Snorlax₩Snorlax.37.jpg
noise
51
Snorlax.56.jpg
D:₩swproject₩pocketmon_classi₩Snorlax₩Snorlax.56.jpg
```

Squirtle 데이터셋 6000개 늘리기 (꼬부기)

In [25]:
```
num_augmented_images = 6000
file_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Squirtle₩₩'
file_names = os.listdir(file_path)
total_origin_image_num = len(file_names)
augment_cnt = 1

#im = Image.open("pocketmon_set3/Squirtle/*")
#rgb_im = im.convert('RGB')
#rgb_im.save('jjajung.jpg')
```

In [26]:

```
for i in range(1, num_augmented_images):
    try:
        change_picture_index = random.randrange(1, total_origin_image_num-1)
        print(change_picture_index)
        print(file_names[change_picture_index])
        file_name = file_names[change_picture_index]

        origin_image_path = 'D:₩₩swproject₩₩pocketmon_classi₩₩Squirtle₩₩' + file_name
        print(origin_image_path)
        image = Image.open(origin_image_path)
        random_augment = random.randrange(1,4)

        if(random_augment == 1):
            #이미지 좌우 반전
            print("invert")
            inverted_image = image.transpose(Image.FLIP_LEFT_RIGHT)
            inverted_image.save(file_path + 'inverted_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 2):
            #이미지 기울이기
            print("rotate")
            rotated_image = image.rotate(random.randrange(-20, 20))
            rotated_image.save(file_path + 'rotated_' + str(augment_cnt) + '.jpg')

        elif(random_augment == 3):
            #노이즈 추가하기
            img = cv2.imread(origin_image_path)
            print("noise")
            row,col,ch= img.shape
            mean = 0
            var = 0.1
            sigma = var**0.5
            gauss = np.random.normal(mean,sigma,(row,col,ch))
            gauss = gauss.reshape(row,col,ch)
            noisy_array = img + gauss
            noisy_image = Image.fromarray(np.uint8(noisy_array)).convert('RGB')
            noisy_image.save(file_path + 'noiseAdded_' + str(augment_cnt) + '.jpg')

        augment_cnt += 1
    except:
        pass
```

```
Squirtle.20.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.20.jpg
rotate
146
Squirtle.230.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.230.jpg
invert
59
Squirtle.152.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.152.jpg
noise
176
Squirtle.258.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.258.jpg
rotate
150
Squirtle.234.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.234.jpg
rotate
204
```

사용할 포켓몬이 들어있는 각각의 폴더의 모든 이름을 변경 (ex). 파이리.1, 파이리.2, 파이리.3 ...)

In [42]:
```
from IPython.display import Image

image_list = os.listdir('pocketmon_classi/') #경로에 있는 파일을 리스트로 생성
print(image_list)
len_image = len(image_list) # image_list길이


for i in image_list: #dataset아래에 있는 폴더명들을 하나씩 i로 가져오기
    file_path_i = 'pocketmon_classi' + '/' + i + '/' #해당 폴더/파일들을 가져오는 경로를 변수에 저장
#     print(file_path_i)
    file_name_i = os.listdir(file_path_i) #파일을 리스트로 저장
#     print(file_name_i)
    j = 1
    for name in file_name_i: #파일하나씩 name변수에 저장
        src = os.path.join(file_path_i, name) # 파일경로랑 name을 연결
        dst = i + '.' + str(j) + '.jpg' #name을 받아서 이름에 번호 붙이기
        dst = os.path.join(file_path_i, dst) #파일경로랑 이름붙인파일명 연결
        os.rename(src, dst) #파일명 변경
#         print(file_name_i)
        j+=1

# print(file_name_i)
```

```
['Charmander', 'Diglett', 'Ditto', 'Eevee', 'Gyarados', 'Meowth', 'Pikachu', 'Rattata', 'Snorlax', 'Squirtle']
```

총 이미지 개수 출력하고, class_name 리스트 출력

In [308]:
```
image_datas = glob("pocketmon_classi/*/*.jpg")
print('Total image:', len(image_datas))
class_name = image_list
class_len = len(class_name)
print(class_name)
```

```
Total image: 60143
['Charmander', 'Diglett', 'Ditto', 'Eevee', 'Gyarados', 'Meowth', 'Pikachu', 'Rattata', 'Snorlax', 'Squirtle']
```

전체 데이터셋 이미지 불러오기

In [310]:
```
image_datas = glob('C:₩₩Users₩₩82106₩₩Desktop₩₩software₩₩pocketmon_all60000₩₩*.jpg')
class_name = ['Eevee', 'Gyarados', 'Meowth', 'Pikachu', 'Rattata', 'Snorlax', 'Squirtle', 'Diglett', 'Ditto', 'Charmander']
dic = {'Eevee':0, 'Gyarados':1, 'Meowth':2, "Pikachu":3, "Rattata":4, 'Snorlax':5, 'Squirtle':6, 'Diglett':7, 'Ditto':8, 'Charmande
```

이미지, 레이블을 저장하기

In [311]:
```
imagename.split('₩₩')
```

Out[311]:
```
['C:',
 'Users',
 '82106',
 'Desktop',
 'software',
 'pocketmon_all60000',
 'Squirtle.999.jpg']
```

In [312]:
```
#데이터들을 담을 리스트 정의
X = list()
#레이블들을 담을 리스트 정의
Y = list()


for imagename in image_datas:
    try:
        image = cv2.imread(imagename)
        image = cv2.resize(image, dsize=(28, 28))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        image = np.array(image)
        X.append(image)

        label = imagename.split('WW')
        label = label[6]
        label = label.split('.')
        label = str(label[0])
        label = dic[label]
        Y.append(label)
    except :   # 예외
        pass

# X, Y리스트들을 NP형식의 배열로 생성
X = np.array(X)
Y = np.array(Y)
print(X)
print(Y)
print('X shape:', X.shape)
print('Y shape:', Y.shape)
```

```
   [254 254 254]]

  [[255 255 255]
   [254 254 254]
   [254 254 254]
   ...
   [254 255 255]
   [255 253 254]
   [252 255 253]]

  [[255 255 255]
   [251 255 254]
   [255 253 254]
   ...
   [252 255 255]
   [255 255 255]
   [255 255 255]]]]
[9 9 9 ... 6 6 6]
X shape: (58378, 28, 28, 3)
Y shape: (58378,)
```

train, test set 나누기

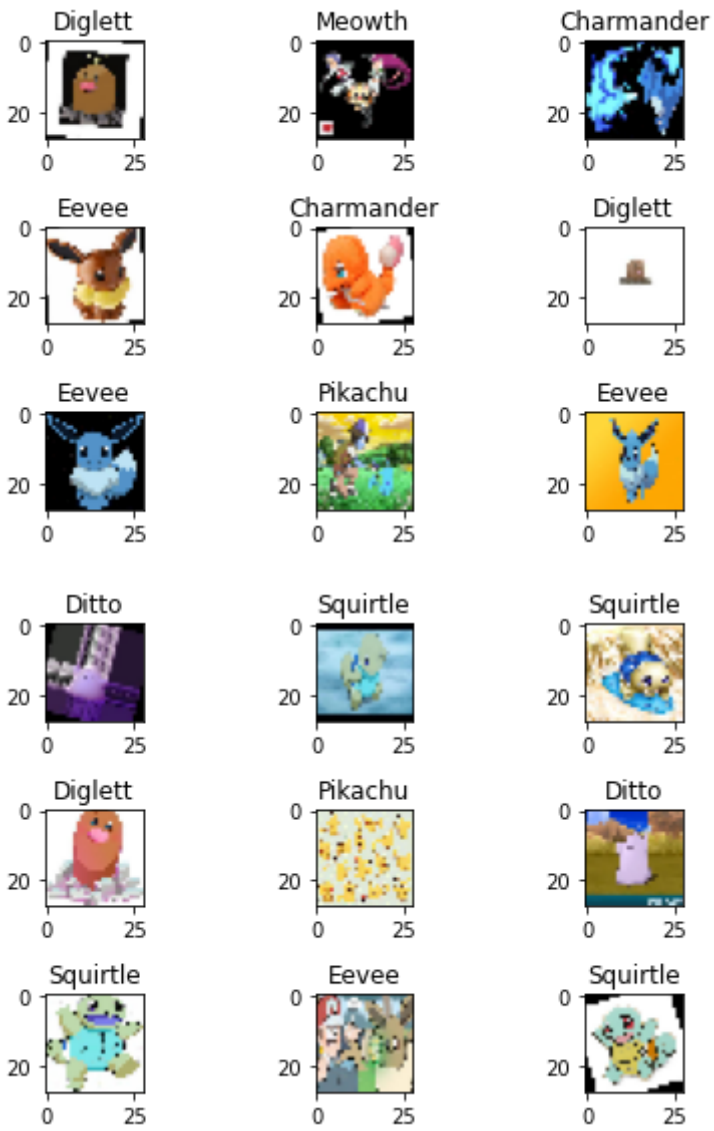
In [317]:
```
train_images, test_images, train_labels, test_labels = train_test_split(X, Y, test_size = 0.2, shuffle=True, random_state=44
print(train_images.shape)
print(test_images.shape)
print(train_labels.shape)
print(test_labels.shape)
```

```
(46702, 28, 28, 3)
(11676, 28, 28, 3)
(46702,)
(11676,)
```

시각화

In [318]:
```
plt.figure()
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(train_images[i])
    tr_po = train_labels[i]
    plt.title(dic2[tr_po])
plt.tight_layout()

plt.figure()
for i in range(9):
    te_po = test_labels[i]
    plt.subplot(3,3,i+1)
    plt.imshow(test_images[i])
    plt.title(dic2[te_po])
plt.tight_layout()
```



정규화

In [319]:
```
L, W, H, C = train_images.shape
train_images = train_images.reshape(-1, H * W * C)
test_images = test_images.reshape(-1, H * W * C)
train_images = train_images.astype('float') / 255
test_images = test_images.astype('float') / 255

print('train_images_shape: ', train_images.shape)
print('test_images_shape: ', test_images.shape)
print(train_images[:5])
print(test_images[:5])
```

```
train_images_shape:  (46702, 2352)
test_images_shape:  (11676, 2352)
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.00392157 0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.99607843 1.         1.         ... 0.99607843 0.97254902 0.97254902]
 [1.         1.         1.         ... 1.         1.         1.        ]
 [0.87058824 0.92156863 0.85882353 ... 0.88235294 0.91372549 0.8627451 ]]
```

원-핫 인코딩

In [320]:
```
Train_labels = to_categorical(train_labels, 10)      #to_cateogrical 함수를 통해 각 라벨을 원핫인코딩(mnist랑 동일)
Test_labels = to_categorical(test_labels, 10)         #to_cateogrical 함수를 통해 각 라벨을 원핫인코딩(mnist랑 동일)
print('train_labels shape:', train_labels.shape)
print('test_labels shape', test_labels.shape)
```

```
train_labels shape: (46702,)
test_labels shape (11676,)
```

인공지능 모델 설계

In [321]:
```
model = Sequential()
model.add(Dense(512, activation = 'relu',
            input_shape=(2352,),
            ))
model.add(Dense(256, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_16 (Dense)             (None, 512)               1204736
_________________________________________________________________
dense_17 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_18 (Dense)             (None, 10)                2570
=================================================================
Total params: 1,338,634
Trainable params: 1,338,634
Non-trainable params: 0
_________________________________________________________________
```

모델 학습시키기

In [326]:
```
early_stopping = EarlyStopping(monitor = 'val_loss', patience=10, verbose=1)
model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
history = model.fit(train_images, Train_labels, batch_size=100, epochs=40, verbose=1,
                    validation_data=(test_images, Test_labels), callbacks = [early_stopping])
```

```
Train on 46702 samples, validate on 11676 samples
Epoch 1/40
46702/46702 [==============================] - 6s 135us/step - loss: 0.0319 - accuracy: 0.9887 - val_loss: 0.0464 - val_accurac
y: 0.9868
Epoch 2/40
46702/46702 [==============================] - 6s 129us/step - loss: 0.0237 - accuracy: 0.9917 - val_loss: 0.0584 - val_accurac
y: 0.9836
Epoch 3/40
46702/46702 [==============================] - 6s 129us/step - loss: 0.0370 - accuracy: 0.9876 - val_loss: 0.0705 - val_accurac
y: 0.9817
Epoch 4/40
46702/46702 [==============================] - 6s 131us/step - loss: 0.0181 - accuracy: 0.9934 - val_loss: 0.0551 - val_accurac
y: 0.9832
Epoch 5/40
46702/46702 [==============================] - 6s 130us/step - loss: 0.0213 - accuracy: 0.9926 - val_loss: 0.0364 - val_accurac
y: 0.9894
Epoch 6/40
46702/46702 [==============================] - 6s 130us/step - loss: 0.0527 - accuracy: 0.9838 - val_loss: 0.0406 - val_accurac
y: 0.9871
Epoch 7/40
46702/46702 [==============================] - 6s 130us/step - loss: 0.0129 - accuracy: 0.9953 - val_loss: 0.0352 - val_accurac
y: 0.9896
Epoch 8/40
46702/46702 [==============================] - 6s 129us/step - loss: 0.0121 - accuracy: 0.9952 - val_loss: 0.0344 - val_accurac
y: 0.9899
Epoch 9/40
46702/46702 [==============================] - 6s 131us/step - loss: 0.0513 - accuracy: 0.9836 - val_loss: 0.0355 - val_accurac
y: 0.9895
Epoch 10/40
46702/46702 [==============================] - 6s 133us/step - loss: 0.0112 - accuracy: 0.9954 - val_loss: 0.0442 - val_accurac
y: 0.9887
Epoch 11/40
46702/46702 [==============================] - 6s 137us/step - loss: 0.0344 - accuracy: 0.9882 - val_loss: 0.1188 - val_accurac
y: 0.9671
Epoch 12/40
46702/46702 [==============================] - 6s 138us/step - loss: 0.0125 - accuracy: 0.9950 - val_loss: 0.0374 - val_accurac
y: 0.9896
Epoch 13/40
46702/46702 [==============================] - 6s 136us/step - loss: 0.0335 - accuracy: 0.9890 - val_loss: 0.1096 - val_accurac
y: 0.9699
Epoch 14/40
46702/46702 [==============================] - 6s 139us/step - loss: 0.0198 - accuracy: 0.9928 - val_loss: 0.0340 - val_accurac
y: 0.9893
Epoch 15/40
46702/46702 [==============================] - 7s 142us/step - loss: 0.0143 - accuracy: 0.9945 - val_loss: 0.7965 - val_accurac
y: 0.8601
Epoch 16/40
46702/46702 [==============================] - 6s 138us/step - loss: 0.0984 - accuracy: 0.9763 - val_loss: 0.0541 - val_accurac
y: 0.9859
Epoch 17/40
46702/46702 [==============================] - 7s 140us/step - loss: 0.0117 - accuracy: 0.9955 - val_loss: 0.0376 - val_accurac
y: 0.9895
Epoch 18/40
46702/46702 [==============================] - 6s 137us/step - loss: 0.0215 - accuracy: 0.9926 - val_loss: 0.1166 - val_accurac
y: 0.9705
Epoch 19/40
46702/46702 [==============================] - 7s 143us/step - loss: 0.0227 - accuracy: 0.9917 - val_loss: 0.0445 - val_accurac
y: 0.9875
Epoch 20/40
46702/46702 [==============================] - 6s 134us/step - loss: 0.0198 - accuracy: 0.9927 - val_loss: 0.0778 - val_accurac
y: 0.9787
Epoch 21/40
46702/46702 [==============================] - 6s 135us/step - loss: 0.0185 - accuracy: 0.9935 - val_loss: 0.0752 - val_accurac
y: 0.9761
Epoch 22/40
46702/46702 [==============================] - 6s 138us/step - loss: 0.0207 - accuracy: 0.9925 - val_loss: 0.0369 - val_accurac
y: 0.9915
Epoch 23/40
46702/46702 [==============================] - 6s 134us/step - loss: 0.0425 - accuracy: 0.9871 - val_loss: 0.0504 - val_accurac
y: 0.9866
Epoch 24/40
46702/46702 [==============================] - 7s 142us/step - loss: 0.0088 - accuracy: 0.9966 - val_loss: 0.0386 - val_accurac
y: 0.9911
Epoch 00024: early stopping
```

모델 정확도 살펴보기

In [327]:
```
score = model.evaluate(test_images, Test_labels)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```
11676/11676 [==============================] - 1s 49us/step
Test score: 0.03864499772090945
Test accuracy: 0.9910928606987
```

예측값, 예측과 맞는 값, 예측과 틀린값 구하기

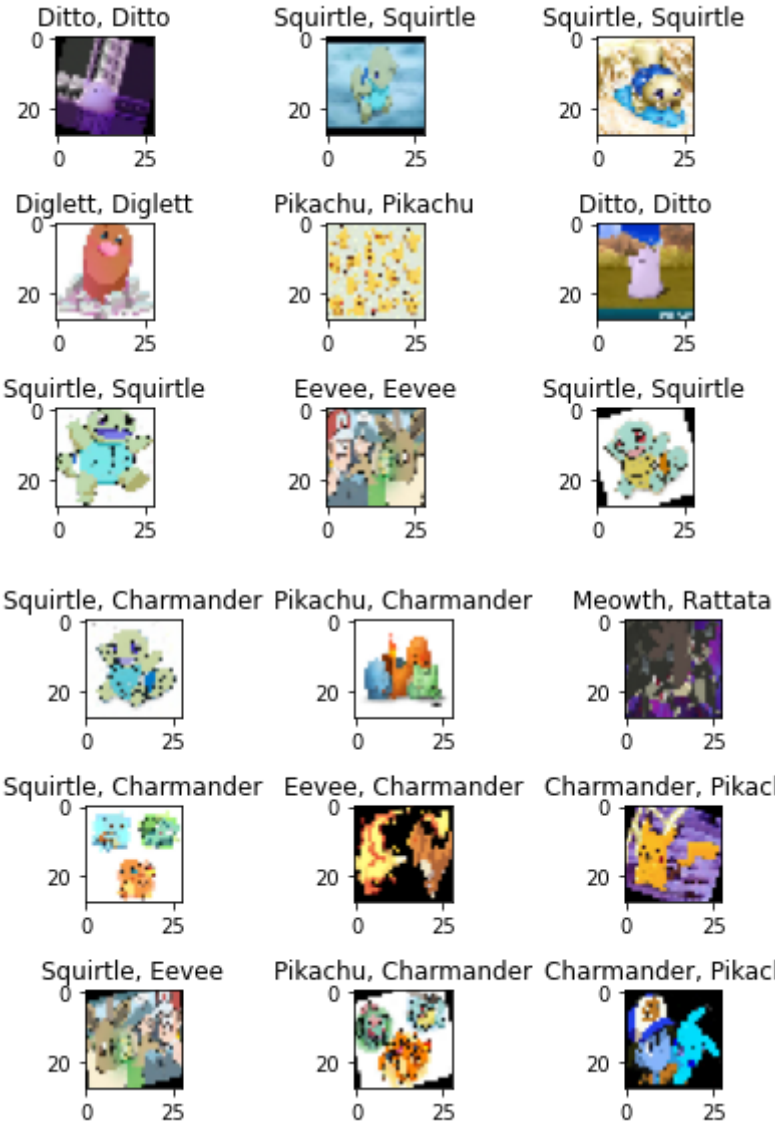
In [328]:
```
predict_classes = np.argmax(model.predict(test_images), axis = 1)
correct_indices = np.nonzero(predict_classes == test_labels)[0]
incorrect_indices = np.nonzero(predict_classes != test_labels)[0]
print(predict_classes)
print(correct_indices)
print(incorrect_indices)
```

```
[8 6 6 ... 2 1 3]
[    0     1     2 ... 11673 11674 11675]
[   84   170   259   311   643   729   739   771   859   901   996  1047
  1115  1118  1427  1461  1681  1682  1687  1757  1770  1927  2006  2102
  2216  2473  2600  2681  2716  2728  2804  2991  3286  3317  3402  3428
  3485  3540  3609  3652  3741  3781  4010  4053  4219  4300  4333  4386
  4391  4478  4520  4566  4639  4901  4921  4975  5120  5178  5259  5306
  5416  5442  5533  5614  5876  5883  6074  6104  6199  6372  6377  6402
  6706  6727  6762  6898  6971  7034  7152  7169  7481  7711  7865  7946
  8074  8271  8387  8896  9280  9634  9923 10072 10413 10454 10528 10589
 10709 10811 11054 11169 11266 11392 11530 11626]
```

위: 예측한 값이랑 실제값이 같은 것을 보여줌, 아래: 예측값과 실제값이 다른 것을 보여줌

In [329]:
```
plt.figure()
for i in range(9):
    plt.subplot(3, 3, i+1)
    correct = correct_indices[i]
    pred1 = predict_classes[correct]
    cla1 = test_labels[correct]
    plt.imshow(test_images[correct].reshape(28, 28, 3))
    plt.title('{}, {}'.format(dic2[pred1], dic2[cla1]))
plt.tight_layout()

plt.figure()
for i in range(9):
    plt.subplot(3, 3, i+1)
    incorrect_label = incorrect_indices[i]
    cla2 = test_labels[incorrect_label]
    pred2 = predict_classes[incorrect_label]
    plt.imshow(test_images[incorrect_label].reshape(28, 28, 3))
    plt.title('{}, {}'.format(dic2[pred2], dic2[cla2]))
plt.tight_layout()
```

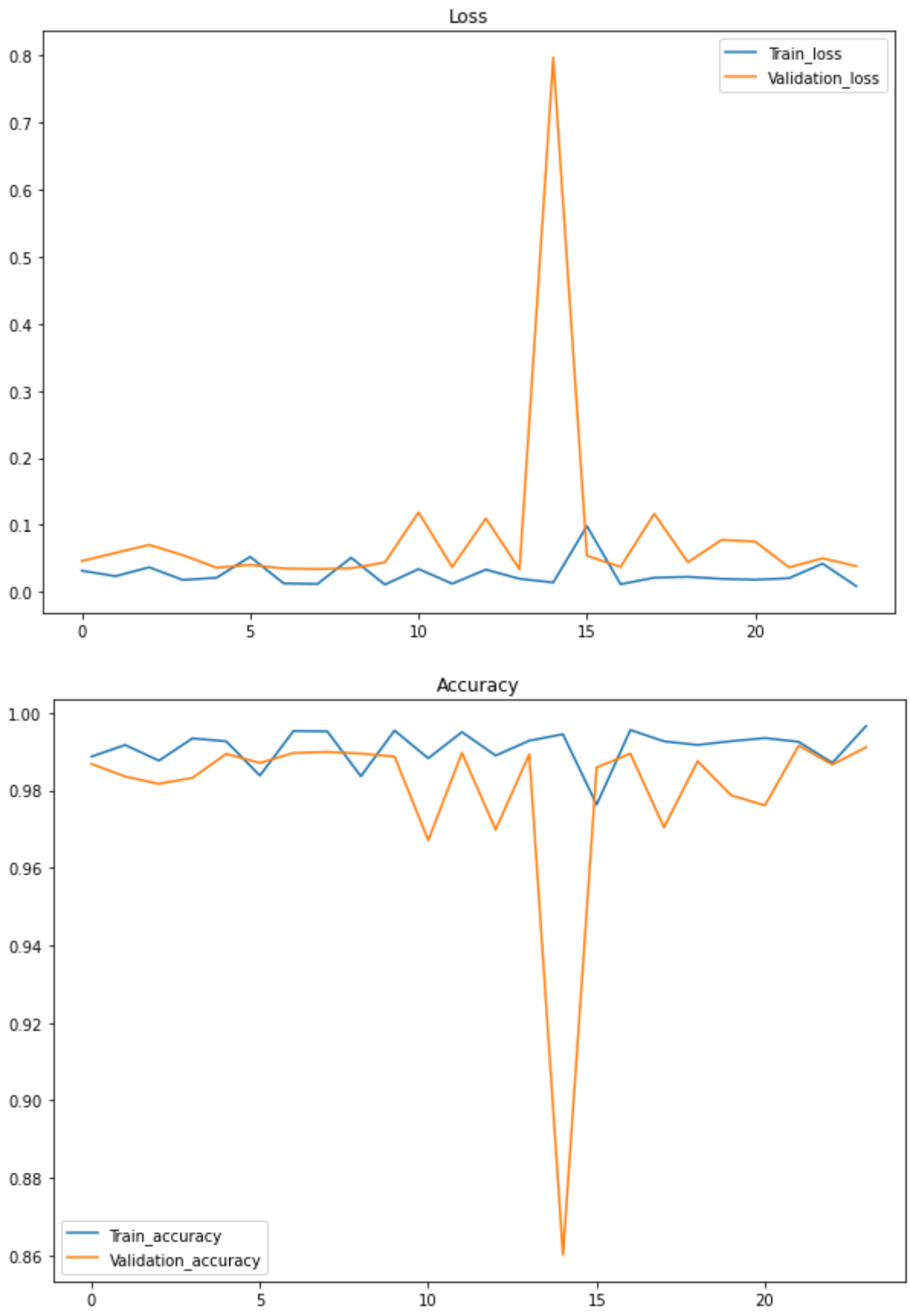


위: Train 데이터를 넣었을 때 오차와 Test 데이터 넣었을 때 오차 시각화 아래: Train 데이터를 넣었을 때 정확도와 Test 데이터 넣었을 때 정확도 시각화

In [330]:
```
plt.figure(figsize=(10,7))
plt.plot(history.history['loss'], label = 'Train_loss')
plt.plot(history.history['val_loss'], label='Validation_loss')
plt.title('Loss')
plt.legend()
plt.show

plt.figure(figsize=(10,7))
plt.plot(history.history['accuracy'], label = 'Train_accuracy')
plt.plot(history.history['val_accuracy'], label='Validation_accuracy')
plt.title('Accuracy')
plt.legend()
plt.show
```

Out[330]: <function matplotlib.pyplot.show(close=None, block=None)>



포켓몬 대결 (두개의 캐릭터를 속성으로 대결하고, 속성이 같으면 hp * 공격력 * 방어력을 이용해 대결해서 대결 승자를 나타내기)

['Charmander:파이리', 'Diglett:디그다', 'Ditto:메타몽', 'Eevee:이브이', 'Gyarados:갸라도스', 'Meowth:나옹', 'Pikachu:피카츄', 'Rattata:꼬렛', 'Snorlax:잠만보', 'Squirtle:꼬부기']

In [172]:
```
# Type: 타입, Hp: hp, Attack: 공격력, Defense: 방어력
# 출처: https://pokemon.fandom.com/ko/wiki/%EC%A2%85%EC%A1%B1%EA%B0%92_%EB%AA%A9%EB%A1%9D
# 출처: https://www.pokemonkorea.co.kr/pokedex/view/193?word=&characters=&area=&snumber=1&snumber2=898&typetextcs=&sortselval
```

데이터프레임 생성

In [249]:
```
image_datas = glob('C:₩₩Users₩₩82106₩₩Desktop₩₩software₩₩pocketmon_all60000₩₩*.jpg')


df = pd.DataFrame([[0,'Eevee'], [1,'Gyarados'], [2,'Meowth'], [3,'Pikachu'], [4,'Rattata'],
                   [5,'Snorlax'], [6,'Squirtle'], [7,'Diglett'], [8,'Ditto'], [9,'Charmander']],
                    columns=['LABEL', 'POCKETMON'])

Type = ['normal', 'water', 'normal', 'electricity', 'normal', 'normal', 'water', 'earth', 'normal', 'fire']
Hp = [55, 95, 40, 35, 39, 160, 44, 10, 48, 39]
Attack = [55, 125, 45, 55, 56, 110, 48, 55, 48, 52]
Defense = [50, 69, 35, 40, 35, 65, 65, 25, 58, 43]

df['TYPE'] = [Type[0], Type[1], Type[2], Type[3], Type[4], Type[5], Type[6], Type[7], Type[8], Type[9]]
df['HP'] = [Hp[0], Hp[1], Hp[2], Hp[3], Hp[4], Hp[5], Hp[6], Hp[7], Hp[8], Hp[9]]
df['ATTACK'] = [Attack[0], Attack[1], Attack[2], Attack[3], Attack[4], Attack[5], Attack[6], Attack[7], Attack[8], Attack[9]]
df['DEFENCE'] = [Defense[0], Defense[1], Defense[2], Defense[3], Defense[4], Defense[5], Defense[6], Defense[7], Defense[8],
df

```

Out[249]:

	LABEL	POCKETMON	TYPE	HP	ATTACK	DEFENCE
0	0	Eevee	normal	55	55	50
1	1	Gyarados	water	95	125	69
2	2	Meowth	normal	40	45	35
3	3	Pikachu	electricity	35	55	40
4	4	Rattata	normal	39	56	35
5	5	Snorlax	normal	160	110	65
6	6	Squirtle	water	44	48	65
7	7	Diglett	earth	10	55	25
8	8	Ditto	normal	48	48	58
9	9	Charmander	fire	39	52	43

if문 사용해서 두개의 캐릭터 대결

In [270]:
```
dic_prop = {'earth':1, 'water':2, 'normal':3, 'electricity':4, 'fire':5}

Eevee = df.loc[0]
Gyarados = df.loc[1]
Meowth = df.loc[2]
Pikachu = df.loc[3]
Rattata = df.loc[4]
Snorlax = df.loc[5]
Squirtle = df.loc[6]
Diglett = df.loc[7]
Ditto = df.loc[8]
Charmander = df.loc[9]


def fight(p1, p2):
    p1_type = p1['TYPE']
    p2_type = p2['TYPE']
    p1_hp_atk_dfs = p1['HP'] * p1['ATTACK'] * p1['DEFENCE']
    p2_hp_atk_dfs = p2['HP'] * p2['ATTACK'] * p2['DEFENCE']
    print('{} VS {} 대결, 승자는 ?' .format(p1['POCKETMON'], p2['POCKETMON']))
    time.sleep(3)
    print()

    if dic_prop[p1_type] < dic_prop[p2_type]:
        print('일방적인 경기가 펼쳐집니다!!')
        time.sleep(1.5)
        print()
        print('{} 승리'.format(p2['POCKETMON']))

    elif dic_prop[p1_type] > dic_prop[p2_type]:
        print('일방적인 경기가 펼쳐집니다!!')
        time.sleep(1.5)
        print()
        print('{} 승리'.format(p1['POCKETMON']))

    else:
        print('엄청 치열합니다!!')
        time.sleep(3)
        print()
        if p1_hp_atk_dfs < p2_hp_atk_dfs:
            print('{} 승리'.format(p2['POCKETMON']))
        else:
            print('{} 승리'.format(p1['POCKETMON']))
```

데이터를 직접 입력해서 포켓몬 대결

In [302]:
```
1 fight(Pikachu, Diglett)
```

```
Pikachu VS Diglett 대결, 승자는 ?

일방적인 경기가 펼쳐집니다!!

Pikachu 승리
```

random으로 두개의 사진을 가져와 화면에 띄우고 무슨 캐릭터인지 예측하고 예측한 두 개의 캐릭터를 대결

In [316]:
```
1 image_list2 = os.listdir('pocketmon_all60000/')
2 choiceList = [random.choice(image_list2) for i in range(2)]
3 print(choiceList)
```

```
['Gyarados.494.jpg', 'Diglett.933.jpg']
```