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 [1]:

```
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
```

```
Using TensorFlow backend.
```

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]:
```python
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
```

```
7
Charmander.105.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.105.jpg
noise
155
Charmander.239.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.239.jpg
noise
152
Charmander.236.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.236.jpg
noise
57
Charmander.150.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.150.jpg
invert
164
Charmander.247.jpg
D:₩swproject₩pocketmon_classi₩Charmander₩Charmander.247.jpg
```

Digrett 데이터셋 6000개 늘리기 (디그다)

In [8]:
```python
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.50.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.50.jpg
invert
19
Diglett.27.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.27.jpg
rotate
18
Diglett.26.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.26.jpg
noise
18
Diglett.26.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.26.jpg
invert
5
Diglett.14.jpg
D:₩swproject₩pocketmon_classi₩Diglett₩Diglett.14.jpg
invert
31
```

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
```

```
invert
39
Ditto.45.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.45.jpg
rotate
26
Ditto.33.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.33.jpg
invert
1

Ditto.10.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.10.jpg
rotate
24
Ditto.31.jpg
D:₩swproject₩pocketmon_classi₩Ditto₩Ditto.31.jpg
invert
25
Ditto.32.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.17.jpg
invert
31
Eevee.38.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.38.jpg
noise
29
Eevee.36.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.36.jpg
invert
15
Eevee.23.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.23.jpg
noise
33
Eevee.4.jpg
D:₩swproject₩pocketmon_classi₩Eevee₩Eevee.4.jpg
rotate
33
```

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
```

```
Gyarados.4.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.4.jpg
rotate
48
Gyarados.53.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.53.jpg
rotate
52
Gyarados.57.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.57.jpg
invert
16
Gyarados.24.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.24.jpg
invert
16
Gyarados.24.jpg
D:₩swproject₩pocketmon_classi₩Gyarados₩Gyarados.24.jpg
invert
21
```

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
```

```
37
Rattata.43.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.43.jpg
invert
50
Rattata.55.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.55.jpg
rotate
45
Rattata.50.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.50.jpg
invert
31
Rattata.38.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.38.jpg
noise
53
Rattata.58.jpg
D:₩swproject₩pocketmon_classi₩Rattata₩Rattata.58.jpg
```

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
```

```
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.118.jpg
noise
248
Squirtle.70.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.70.jpg
rotate
32
Squirtle.128.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.128.jpg
rotate
174
Squirtle.256.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.256.jpg
noise
106
Squirtle.195.jpg
D:₩swproject₩pocketmon_classi₩Squirtle₩Squirtle.195.jpg
noise
150
Squirtle.234.jpg
```

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

In [4]:
```python
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 [5]:
```python
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 [10]:
```python
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, 'Charmander':9}
dic2 = {0:'Pikachu', 1:'Bulbasaur', 2:'Psyduck', 3:'Clefairy', 4:'Gastly', 5:'Growlithe',
        6:'Jigglypuff', 7:'Mew', 8:'Poliwag', 9:'Slowpoke'}
```

이미지, 레이블을 저장하기

In [7]:
```
#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('₩₩')
        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)
```

```
[[[[  0   0   0]
   [  0   0   0]
   [  0   0   0]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

  [[  0   0   0]
   [  0   0   0]
   [  0   0   0]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

  [[  0   0   0]
   [  0   0   0]
   [  0   0   0]
```

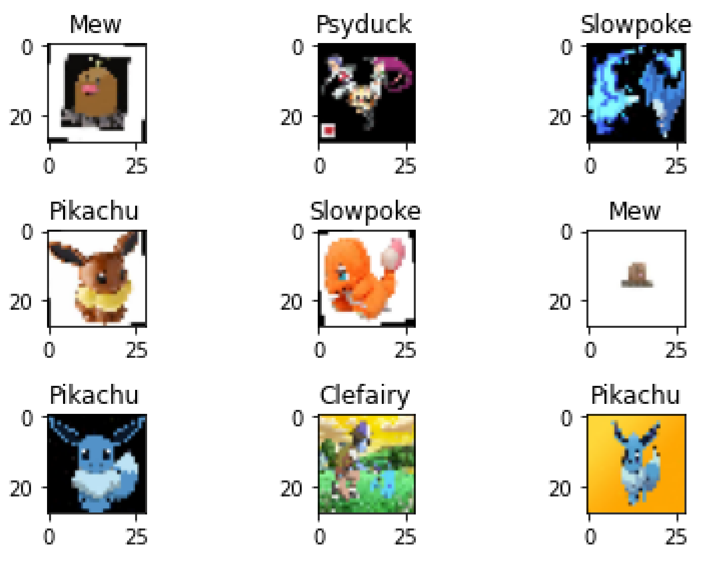
train, test set 나누기

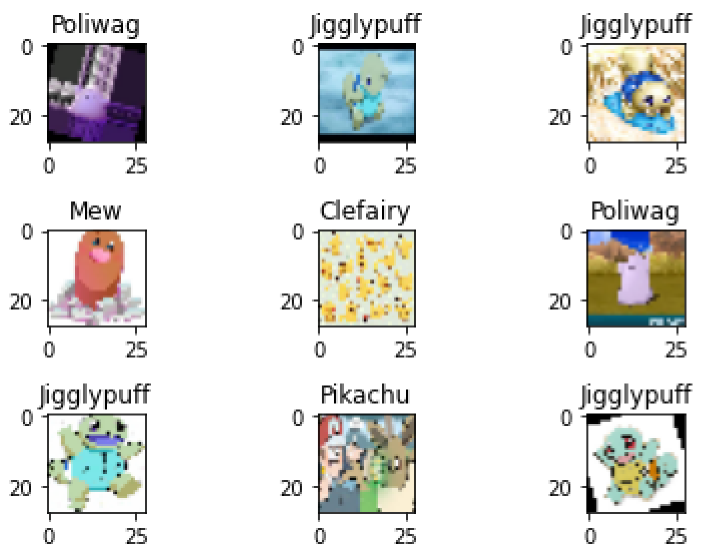
In [8]:
```
train_images, test_images, train_labels, test_labels = train_test_split(
    X, Y, test_size = 0.2, shuffle=True, random_state=44)
#train_test_split 함수를 사용하여 훈련_이미지, 테스트_이미지, 훈련_라벨, 테스트_라벨을 8:2로 나눔
print(train_images.shape)
print(test_images.shape)
print(train_labels.shape)
print(test_labels.shape)
```

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

시각화

In [11]:
```python
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 [12]:
```python
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 [13]:
```
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 [14]:
```
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_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               1204736
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,338,634
Trainable params: 1,338,634
Non-trainable params: 0
_________________________________________________________________
```

모델 학습시키기

In [15]:
```
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])
```

```
acy: 0.8972
Epoch 7/40
46702/46702 [==============================] - 7s 149us/step - loss: 0.1266 - accuracy: 0.9599 - val_loss: 0.1706 - val_accur
acy: 0.9409
Epoch 8/40
46702/46702 [==============================] - 7s 151us/step - loss: 0.1041 - accuracy: 0.9662 - val_loss: 0.3522 - val_accur
acy: 0.8881
Epoch 9/40
46702/46702 [==============================] - 7s 150us/step - loss: 0.1040 - accuracy: 0.9658 - val_loss: 0.1078 - val_accur
acy: 0.9659
Epoch 10/40
46702/46702 [==============================] - 7s 152us/step - loss: 0.0834 - accuracy: 0.9740 - val_loss: 0.1446 - val_accur
acy: 0.9508
Epoch 11/40
46702/46702 [==============================] - 7s 152us/step - loss: 0.0685 - accuracy: 0.9779 - val_loss: 0.1091 - val_accur
acy: 0.9667
Epoch 12/40
46702/46702 [==============================] - 7s 152us/step - loss: 0.1020 - accuracy: 0.9671 - val_loss: 0.1029 - val_accur
acy: 0.9689
Epoch 13/40
```

모델 정확도 살펴보기

In [16]:
```
score = model.evaluate(test_images, Test_labels)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```
11676/11676 [==============================] - 1s 56us/step
Test score: 0.04799433251530661
Test accuracy: 0.9868962168693542
```

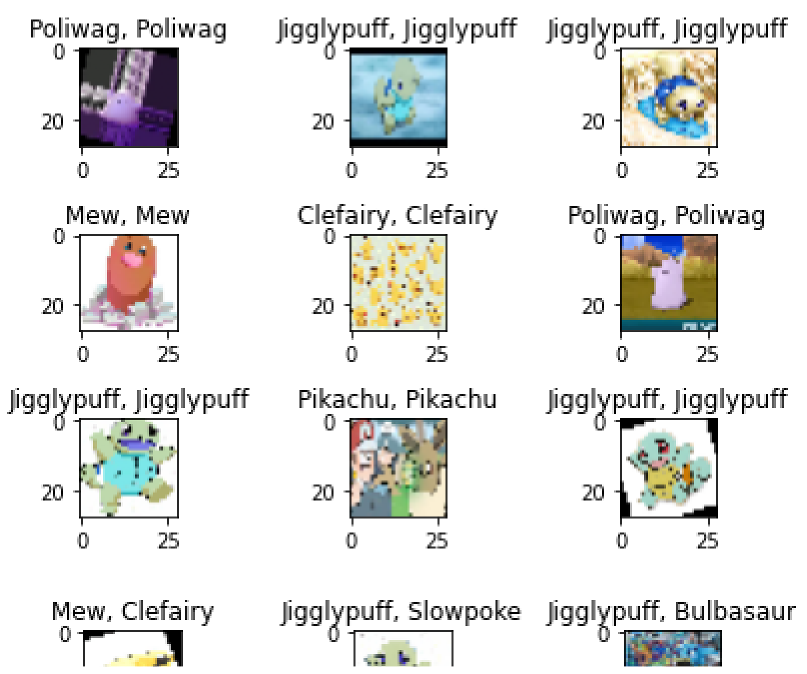
예측값, 예측과 맞는 값, 예측과 틀린값 구하기

In [17]:
```python
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]
[   33    84   160   209   259   311   362   419   540   583   601   643
   695   729   739   771   901   908  1047  1115  1118  1290  1303  1357
  1369  1427  1429  1666  1757  1927  2120  2471  2473  2488  2600  2804
  2866  3090  3141  3214  3264  3286  3300  3402  3485  3540  3564  3574
  3629  3736  3741  3748  3758  3781  3888  4010  4067  4092  4333  4386
  4391  4520  4549  4566  4921  4971  4975  5065  5096  5103  5112  5120
  5178  5244  5306  5309  5314  5324  5442  5533  5572  5614  5830  5876
  5883  5937  6074  6199  6313  6318  6372  6377  6402  6407  6543  6611
  6727  6762  6898  6971  7034  7169  7204  7358  7481  7529  7711  7744
  7820  7865  7867  7943  8016  8343  8387  8440  8464  8925  9022  9094
  9180  9280  9373  9388  9634  9653  9725  9783  9907  9923  9936 10114
 10200 10237 10454 10514 10589 10802 10811 10860 10876 10942 10979 11054
 11089 11169 11171 11315 11369 11530 11602 11626 11654]
```

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

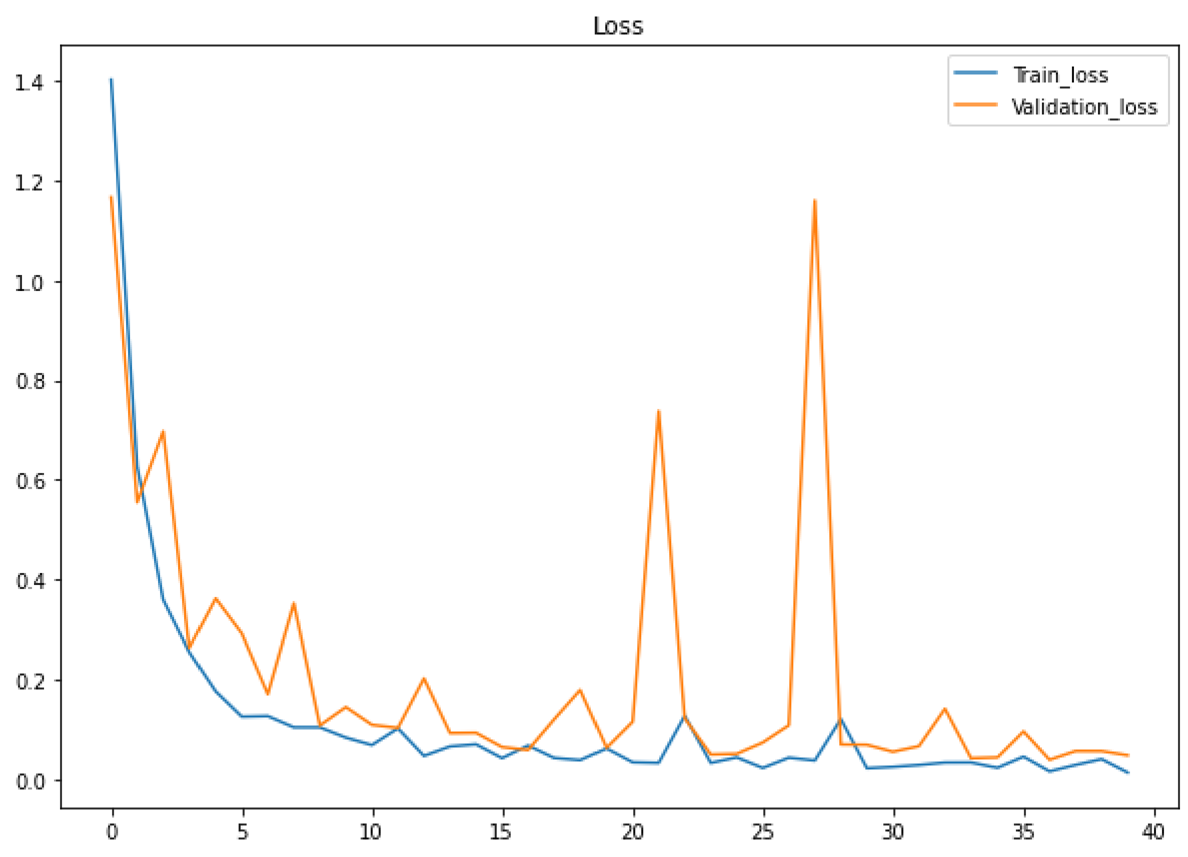
In [18]:
```python
plt.figure()
for i in range(9):
    plt.subplot(3, 3, i+1)
    correct = correct_indices[i]
    pred1 = predict_classes[correct]
    cla1 = test_labels[correct]
    plt.imshow(test_images[correct].reshape(28, 28, 3))
    plt.title('{}, {}'.format(dic2[pred1], dic2[cla1]))
plt.tight_layout()

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()
```

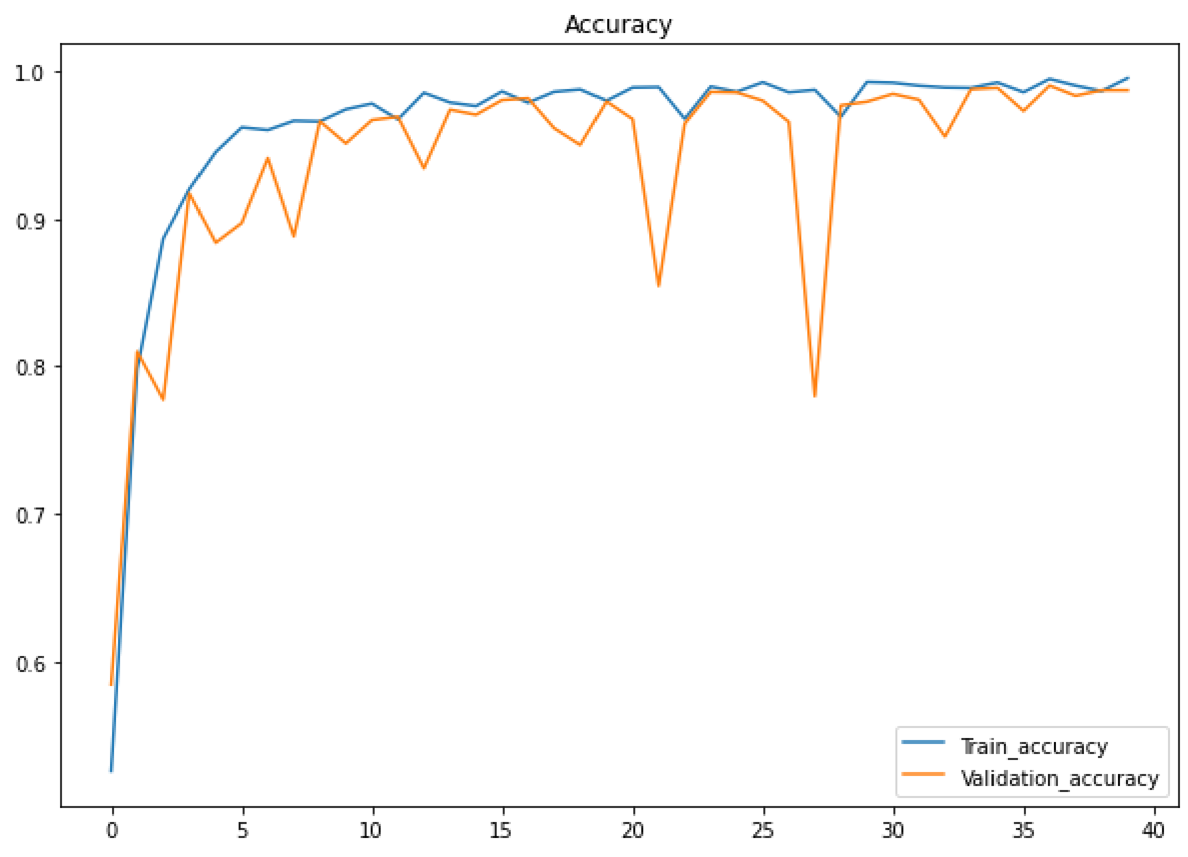


위: Train 데이터를 넣었을 때 오차와 Test 데이터 넣었을 때 오차 시각화 아래: Train 데이터를 넣었을 때 정확도와 Test 데이터 넣었을 때 정확도 시각화

In [19]:
```python
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[19]:
```
<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 [20]:
```
# 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 [22]:
```
import pandas as pd

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], Defense[9]]
df

```

Out[22]:

	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 [23]:
```
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 [24]:
```
fight(Pikachu, Diglett)
```

```
Pikachu VS Diglett 대결, 승자는 ?

일방적인 경기가 펼쳐집니다!!

Pikachu 승리
```

In [25]:
```
fight(Eevee, Snorlax)
```

```
Eevee VS Snorlax 대결, 승자는 ?

엄청 치열합니다!!

Snorlax 승리
```

random으로 두개의 사진을 가져와 or 입력해 화면에 띄우고 무슨 캐릭터인지 예측하고 예측한 두 개의 캐릭터를 대결 - 실패

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

머신러닝 모델

RandomForest 분류 모델 사용

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
clf = RandomForestClassifier(n_estimators=100)
clf.fit(train_images, train_labels)

pred = clf.predict(test_images)

print("테스트 정확도 :", str(accuracy_score(test_labels, pred)))
print(classification_report(test_labels, pred))
```

```
테스트 정확도 : 0.9945186707776635
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1176
           1       1.00      1.00      1.00      1207
           2       1.00      1.00      1.00      1199
           3       0.99      0.98      0.99      1045
           4       1.00      1.00      1.00      1200
           5       1.00      1.00      1.00      1269
           6       0.98      0.99      0.98      1087
           7       1.00      1.00      1.00      1200
           8       1.00      1.00      1.00      1214
           9       0.99      0.98      0.98      1079

    accuracy                           0.99     11676
   macro avg       0.99      0.99      0.99     11676
weighted avg       0.99      0.99      0.99     11676
```

SupportVector 분류 모델 사용

In [34]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
svc = SVC()
svc.fit(train_images, train_labels)

pred = svc.predict(test_images)

print("테스트 정확도 :", str(accuracy_score(test_labels, pred)))
print(classification_report(test_labels, pred))
```

```
테스트 정확도 : 0.9898937992463173
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1176
           1       1.00      1.00      1.00      1207
           2       0.99      1.00      0.99      1199
           3       0.99      0.98      0.98      1045
           4       0.99      1.00      1.00      1200
           5       1.00      1.00      1.00      1269
           6       0.97      0.97      0.97      1087
           7       0.99      1.00      0.99      1200
           8       1.00      1.00      1.00      1214
           9       0.98      0.95      0.97      1079

    accuracy                           0.99     11676
   macro avg       0.99      0.99      0.99     11676
weighted avg       0.99      0.99      0.99     11676
```

DecisionTree 분류 모델 사용

In [47]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
tree = DecisionTreeClassifier()
tree.fit(train_images, train_labels)

pred = tree.predict(test_images)

print("테스트 정확도 :", str(accuracy_score(test_labels, pred)))
print(classification_report(test_labels, pred))
```

```
테스트 정확도 : 0.9457862281603289
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      1176
           1       0.95      0.93      0.94      1207
           2       0.95      0.94      0.95      1199
           3       0.93      0.92      0.92      1045
           4       0.95      0.95      0.95      1200
           5       0.95      0.96      0.96      1269
           6       0.91      0.89      0.90      1087
           7       0.97      0.98      0.98      1200
           8       0.99      0.99      0.99      1214
           9       0.91      0.90      0.91      1079

    accuracy                           0.95     11676
   macro avg       0.94      0.94      0.94     11676
weighted avg       0.95      0.95      0.95     11676
```

피카츄 폴더 안에서 틀린 데이터 하나를 찾아서 출력해내기

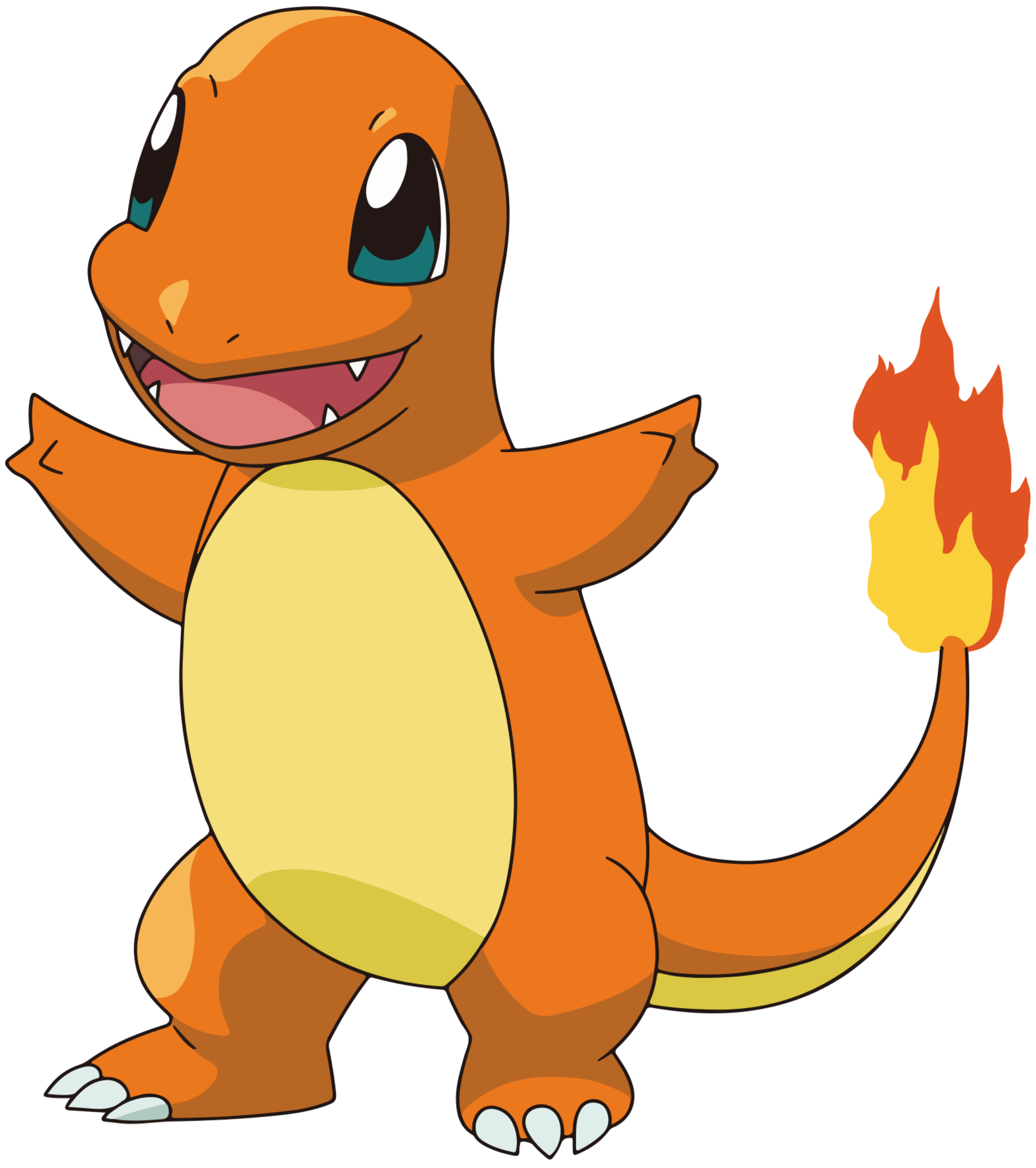
In [75]:
```
image_list_Pika = os.listdir('C:/Users/82106/Desktop/software/pikachu_dda/')
image_list_Pika
```

Out[75]:
```
['Pikachu.1.jpg',
 'Pikachu.10.jpg',
 'Pikachu.2.jpg',
 'Pikachu.3.jpg',
 'Pikachu.4.jpg',
 'Pikachu.5.jpg',
 'Pikachu.6.jpg',
 'Pikachu.7.jpg',
 'Pikachu.8.jpg',
 'Pikachu.9.jpg']
```

In [86]:
```python
df5 = 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'])

# last_e에 해당하는 번호를 딕셔너리 값을 불러오면 포켓몬 이름이 됨
last_d = {0:'Eevee', 1:'Gyarados', 2:'Meowth', 3:'Pikachu', 4:'Rattata', 5:'Snorlax',6:'Squirtle', 7:'Diglett',
          8:'Ditto', 9:'Charmander'}

s = [] # 예측값 넣을 리스트 생성
for i in range (10):
    image_pp = "Pikachu_dda/"+ image_list_Pika[i] # 폴더안의 파일을 처음부터 image_pp에 저장
    predict_i = cv2.imread(image_pp)
    predict_i = cv2.resize(predict_i, dsize=(28,28)) # 이미지 픽셀조정
    predict_i = cv2.cvtColor(predict_i, cv2.COLOR_BGR2RGB) # 이미지 컬러지정
    W, H, C = predict_i.shape
    predict_i = predict_i.reshape(-1, H * W * C)
    predict_i = predict_i.astype('float') / 255
    last_i = np.argmax(model.predict(predict_i), axis=1) # 예측값에 해당하는 딕셔너리의 key값을 받아오기위해 번호 저장
    last_i = int(last_i)
    s.append(last_d[last_i]) #파일 예측값을 처음부터 리스트로 저장

sp= image_pp.split('/')
sp2 = sp[0].split("_")
print('폴더의 주인포켓몬: ', sp2[0])

for i in range(10):
    if s[i] != sp2[0]: # 예측값을 처음부터 label과 비교하여 다른것인 경우
        print("숨어든 범인! : ", s[i])
        result_path = 'Pikachu_dda' + '/' + image_list_Pika[i] #범인의 경로를 저장

Image(result_path) # 범인을 이미지로 출력(단순 출력)
```

```
폴더의 주인포켓몬:  Pikachu
숨어든 범인! :  Charmander
```

Out[86]:



나옹 폴더 안에서 틀린 데이터 하나를 찾아서 출력해내기

In [87]:
```
image_list_Meo = os.listdir('C:/Users/82106/Desktop/software/Meowth_dda/')
image_list_Meo
```

Out[87]:
```
['Meowth.1.jpg',
 'Meowth.10.jpg',
 'Meowth.2.jpg',
 'Meowth.3.jpg',
 'Meowth.4.jpg',
 'Meowth.5.jpg',
 'Meowth.6.jpg',
 'Meowth.7.jpg',
 'Meowth.8.jpg',
 'Meowth.9.jpg']
```

In [90]:
```
df5 = 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'])
last_d = {0:'Eevee', 1:'Gyarados', 2:'Meowth', 3:'Pikachu', 4:'Rattata', 5:'Snorlax',
          6:'Squirtle', 7:'Diglett', 8:'Ditto', 9:'Charmander'}

s = []
for i in range (10):
    image_pp = "Meowth_dda/"+ image_list_Meo[i]
    predict_i = cv2.imread(image_pp)
    predict_i = cv2.resize(predict_i, dsize=(28,28))
    predict_i = cv2.cvtColor(predict_i, cv2.COLOR_BGR2RGB)
    W, H, C = predict_i.shape
    predict_i = predict_i.reshape(-1, H * W * C)
    predict_i = predict_i.astype('float') / 255
    last_i = np.argmax(model.predict(predict_i), axis=1)
    last_i = int(last_i)
    s.append(last_d[last_i]) #파일 예측값 리스트로 저장

sp= image_pp.split('/')
sp2 = sp[0].split("_")
print('폴더의 주인포켓몬: ', sp2[0])

for i in range(10):
    if s[i] != sp2[0]:
        print("숨어든 범인! : ", s[i])
        result_path = 'Meowth_dda' + '/' + image_list_Meo[i]


Image(result_path)
```

```
폴더의 주인포켓몬:  Meowth
숨어든 범인! :  Snorlax
```

Out[90]: