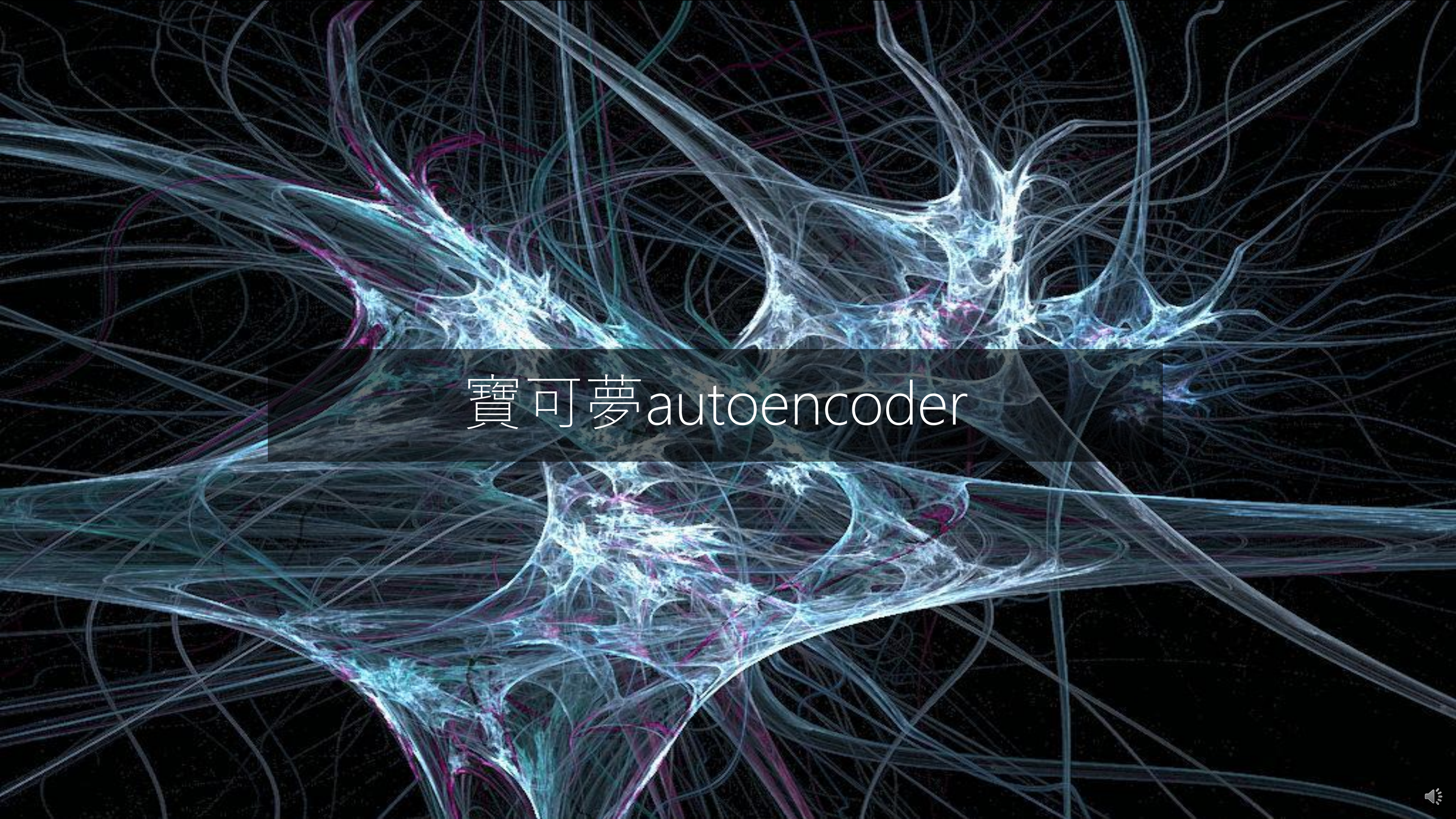
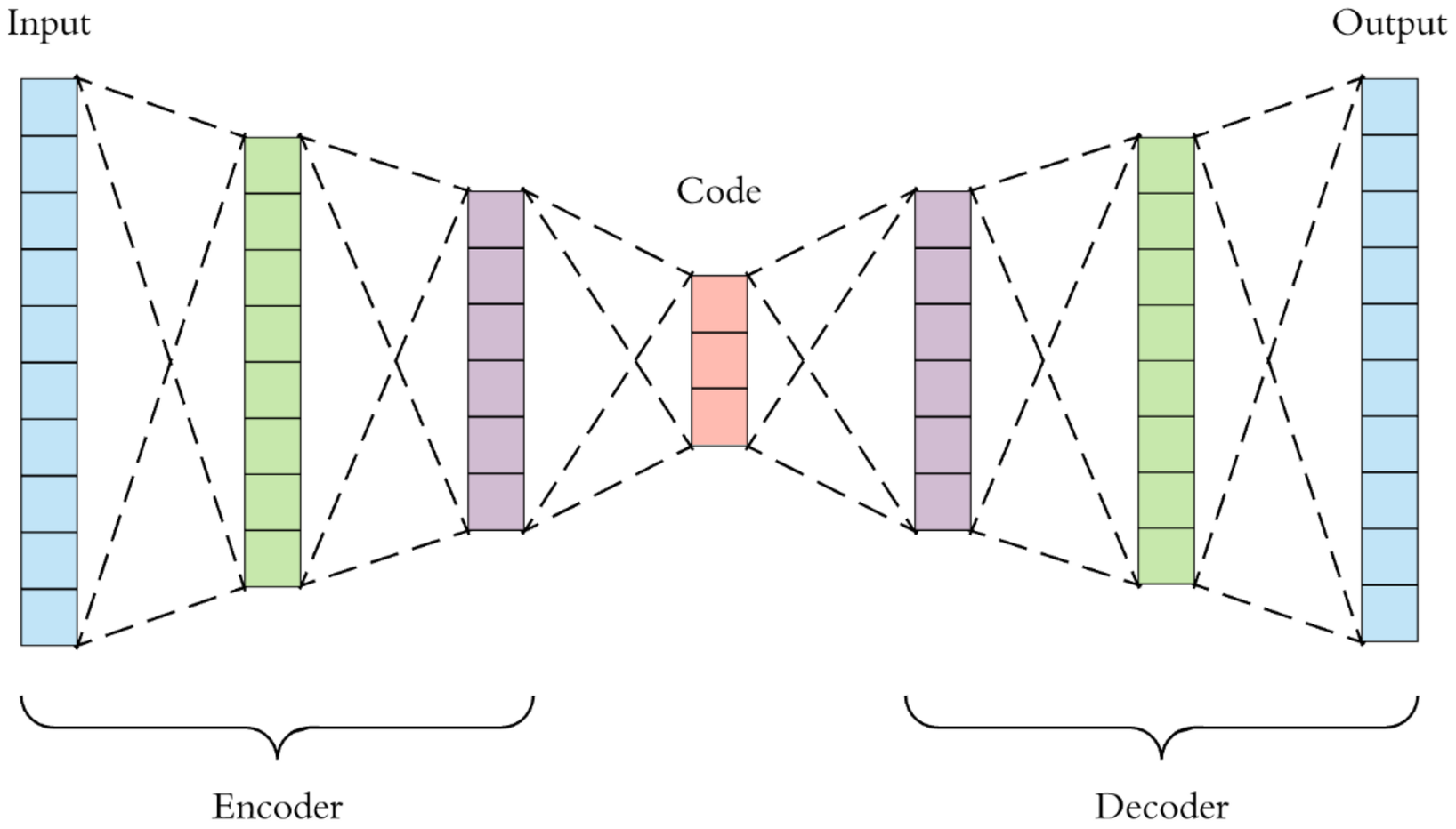

寶可夢autoencoder

實做案例位置
github

https://github.com/AllanYiin/DeepBelief_Course5_Examples/tree/master/prewarm_自動寶可夢編碼器

colab

https://drive.google.com/drive/folders/1i95eyMxrP1QekrcGL7mvHi3U2RVtceoZ?usp=sharing


Input
Output
Code
Encoder
Decoder

File Edit View Insert Cell Kernel Help Python [Anaconda3]

Code CellToolbar

In [1]:
```
%matplotlib inline

import tkinter
import matplotlib
import platform
if platform.system() not in ['Linux', 'Darwin'] and not platform.system().startswith('CYGWIN'):
    matplotlib.use('TKAgg')
import matplotlib.pyplot as plt
from IPython import display
```

寶可夢自動編碼器 (pytorch)

支援python 版本: 3.5以上
支援pytorch版本 : 1.2以上

深度學習的關鍵就是「表徵學習(representation learning)」，透過最佳化算法算更新神經元權重的同時，也正是將所搜尋到的特徵進一步檢視是有用的保留，還是冗餘的捨棄，而autoencoder 自動編碼器正是找尋關鍵特徵並將其充分壓縮的經典網路結構。在這個實作範例中，我們將帶著大家設計一個簡單的卷積自編碼器，而輸入的數據正是目前很流行的寶可夢，我們要來實證看看，光是利用沒有做任何標註的數據，自編碼器是否能夠有效的找出關鍵特徵。

In [2]:
```
import glob
import os
import cv2
os.environ['TRIDENT_BACKEND'] = 'pytorch'
#!pip install tridentx --upgrade
import trident as T
from trident import *
```

所以以上TrainingPlan的設定就是
1. 加入TrainingItem (包含了模型、優化器、損失函數、效度指標、權重正則)
2. 指定 data_loader
3. 重複500 epoch
4. 指定大小
5. 指定學習率變化模式
6. 印出學習進度
7. 指定模型存檔週期
8. 顯示autoencoder 效果輸出圖
9. 指定繪製損失函數與指標變動歷史圖的週期

除了預設的功能外，事實上也可以利用 Callbacks 的機制在關鍵時間點插入自定義工作，這些特性之後也會在介紹，設定完成後，只需要透過start_now()函數即可啟動。

In [ ]:
```
plan.start_now()
```

以下是我跑了 500 epoch 的成果，是不是可以看到從雜訊到模糊，一直到越來越清晰的過程呢，只要訓練的時間夠久就能夠越來越清晰。如果您手邊沒有 gpu ，建議可以透過有免費gpu的google 的colab 來執行。

In [ ]:
```
display.Image('../images/pokemon_training.png',width=800)
```

表徵學習

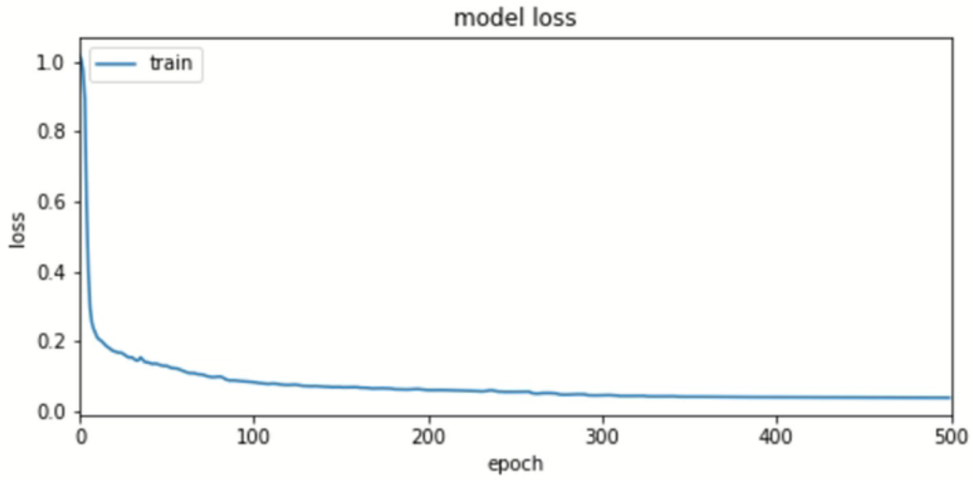
可能會有人覺得奇怪，自編碼器重建圖像就算訓練好了，到底是有甚麼作用？其實，對我們有用的並不是整個自編碼器，我們要的其實是前半段的編碼器部分。編碼器的工作是將圖片編碼成長度為 128 向量(為了避免使用到全連接層，所以實際上是 (128,1,1) 的形狀)，等於是將圖片抽出它的關鍵特徵，而這些特徵既然可以用來還原回圖像細節，這表示它必定包含了這個圖片中的關鍵訊息。這也是深度學習中表徵學習(representation learning)中最常見的手法。而這些特徵向量就可以幫助我們評估圖片中的相似性，也就是可以做到視覺搜索的效果。

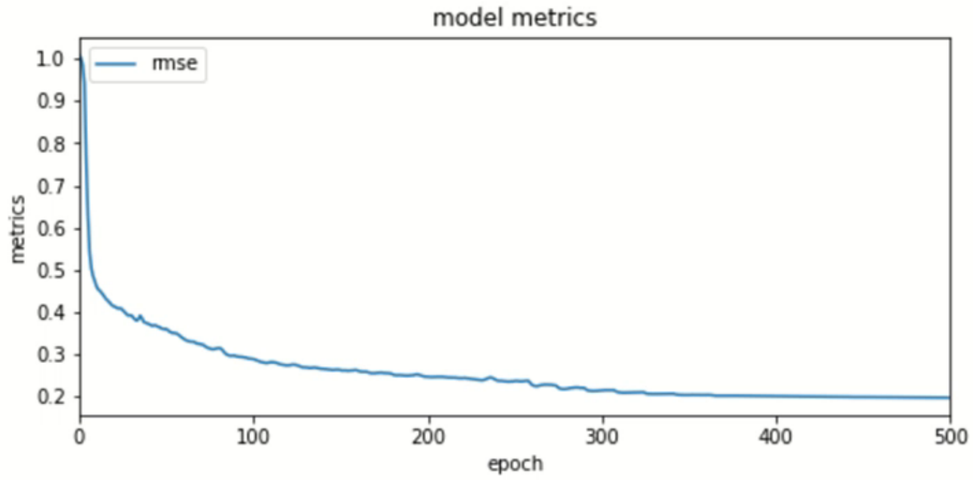
我們首先利用資料源的get_all_data()函數取出所有圖片，依序透過編碼器(autoencoder[0])

File Edit View Insert Cell Kernel Help Python [Anaconda3]

Code CellToolbar

In [9]:
```
plan.start_now()
```





Step: 11s904ms | Tot: 2h15m | Loss: 0.039 | Rmse: 19.773% | learning rate: 2.6697e-04 (500/500)

以下是我跑了 500 epoch 的成果，是不是可以看到從雜訊到模糊，一直到越來越清晰的過程呢，只要訓練的時間夠久就能夠越來越清晰。如果您手邊沒有 gpu ，建議可以透過有免費gpu的google 的colab 來執行。

File Edit View Insert Cell Kernel Help Python [Anaconda3]

CellToolbar

Run Cells
Run Cells and Select Below
Run Cells and Insert Below
Run All
Run All Above
Run All Below
Cell Type
Current Outputs
All Output

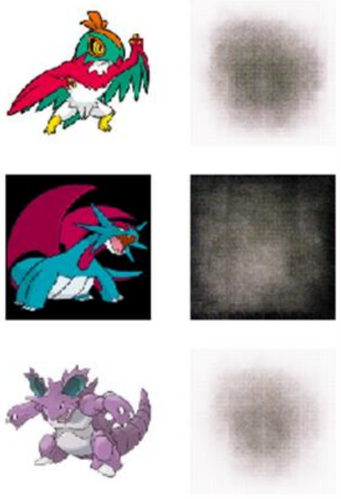
In [ ]:
```
features=[]


#dataset.data['trai
for img_data in dat
  input=to_tensor(                                    e_transform(img_data),0))
  encoder_output                                      ncoder[0](input)))
  features.append(


features=np.asarra
print(features.shap
features=to_tensor(features)
```

我們如果想要看一下特徵向量整體的效果可透過傳說中的降維神器 t-SNE ，將特徵向量降為並且視覺化。

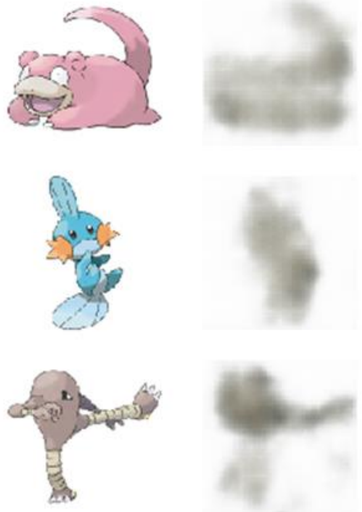
In [ ]:
```
from matplotlib import offsetbox
from sklearn import manifold
import PIL
from PIL import Image as Image

def plot_embedding(X, title=None):
  x_min, x_max = np.min(X, 0), np.max(X, 0)
  X = (X - x_min) / (x_max - x_min)

  fig =plt.figure(figsize=(18,18))
  ax = plt.subplot(111)

  if hasattr(offsetbox, 'AnnotationBbox'):
    # 需要matplotlib 版本> 1.0才支援顯示圖片功能
    shown_images = np.array([[1., 1.]])  # just something big
    for i in range(X.shape[0]):
      dist = np.sum((X[i] - shown_images) ** 2, 1)
```

Minibatch=0

Minibatch=20



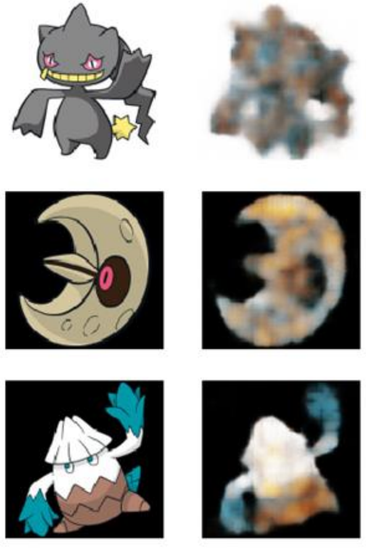
Minibatch=50



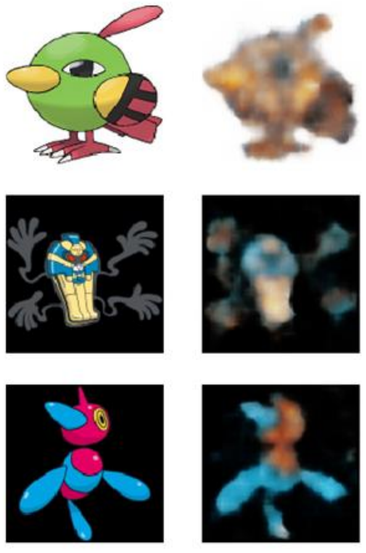
Minibatch=100

Minibatch=200

Minibatch=300



Minibatch=400

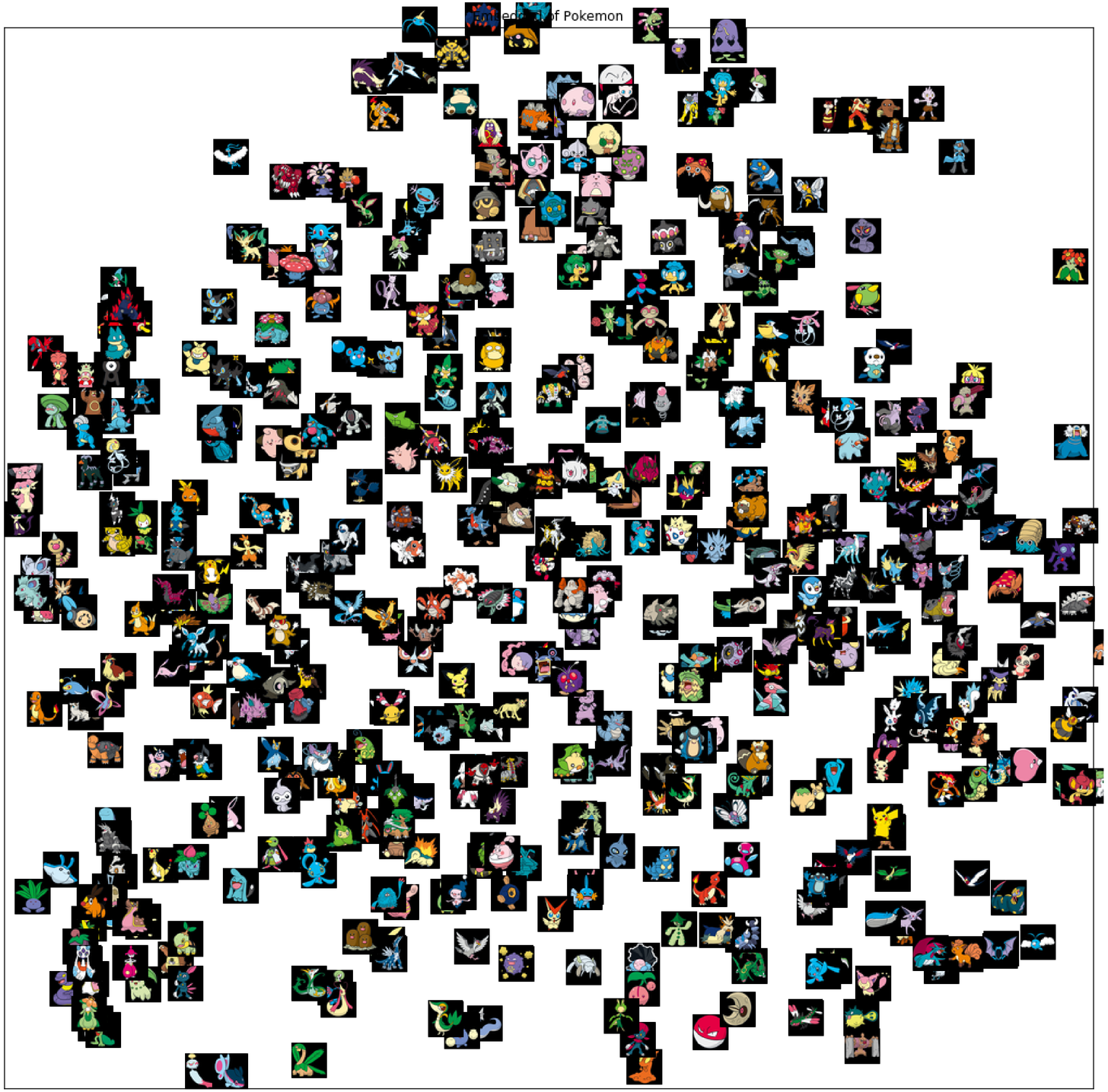


Minibatch=500

寶可夢的相似性搜索

In [12]:
```
idx=10 #抽取一隻寶可夢

similarity_list=[]
result=to_numpy(element_cosine_distance(features[idx:idx+1,:],features))  #element_cosine_distance逐成員計算Cosine距離


top5=np.argsort(result)[-5:][::-1]  #找出前 5個Cosine距離最高者(Cosine距離是越高越像)
similarity_list=[dataset.data['train'][idx]] #放入原圖
similarity_list.append(np.ones_like(similarity_list[0])[:,:30,:]*255) # 加入白色分隔線
similarity_list.extend(dataset.data['train'][top5]) #放入前 5名圖

merge_img=np.concatenate(similarity_list,axis=1) #沿著寬(axis=1)疊合
display.display(array2img(merge_img)) #顯示結果
```

```
[ 0.81406164  0.21352267  0.5302597  ...  0.0266456  -0.15579985
 -0.2514869 ]
```


of Pokemon


Q&A