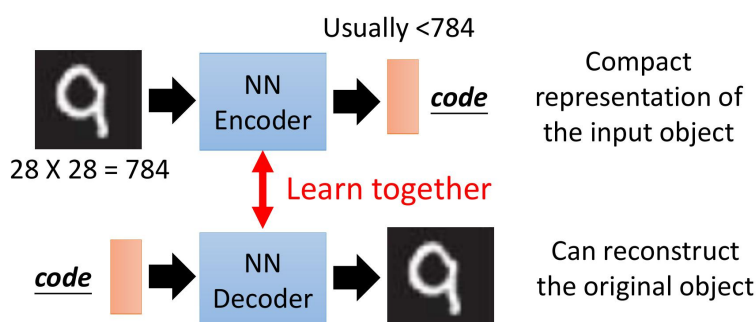


## Auto-encoder

- Unsupervised Learning
- 壓縮的效果
- Encoder
  - From input to vector
- Decoder
  - From vector to origin Input
- Need to train Encoder and Decoder together

### Auto-encoder



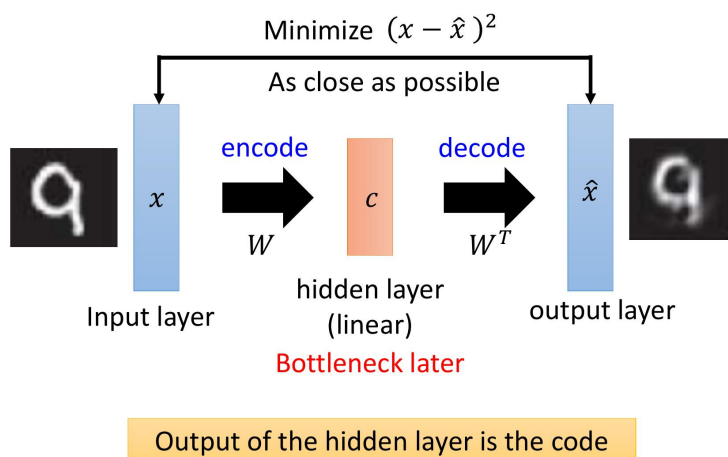
---

## Recap : PCA

- In PCA
  - v.s. In Neural Network
- Minus  $\bar{x}$ 
  - like normalize in neural network (NN)
- Time a weight
  - like connect a layer in NN
- minimize the difference between input and reconstruction
  - make the output like the input
- Input  $x$ 
  - Input layer
- Output 的  $\hat{x}$

- Output layer
- Component's weight
  - bottleneck layer
  - which has lower dimensions

## Recap: PCA

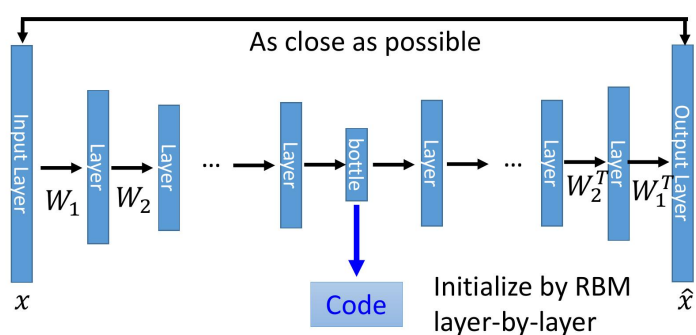


## Deep Auto-encoder

### Deep Auto-encoder

Symmetric is not necessary.

- Of course, the auto-encoder can be deep



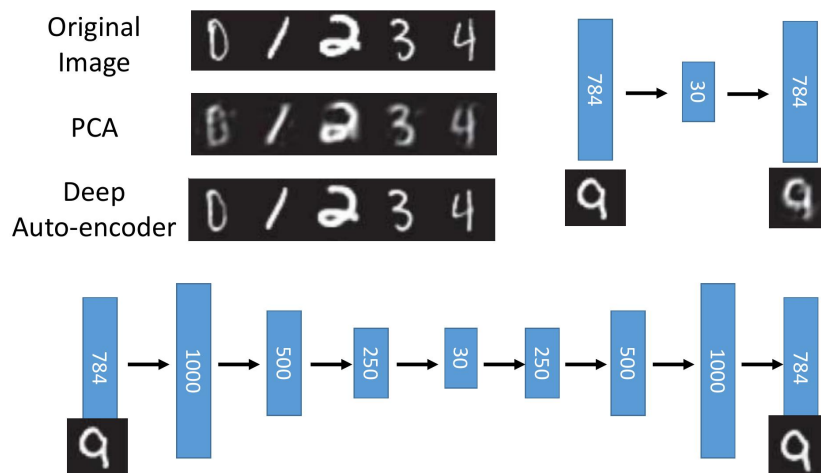
Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

- PCA 只有一個 hidden layer，而我們可以有更多 hidden layer
- output 是  $\hat{x}$ ，希望這個  $x$  跟  $\hat{x}$  越接近越好
- training: back propagation

- 從 input 到 bottleneck layer 的部分就是 encoder
- 從 bottleneck layer 的 output 到最後的  $\hat{x}$  到最後整個 network 的 output 就是 decoder

## Compared on MNIST

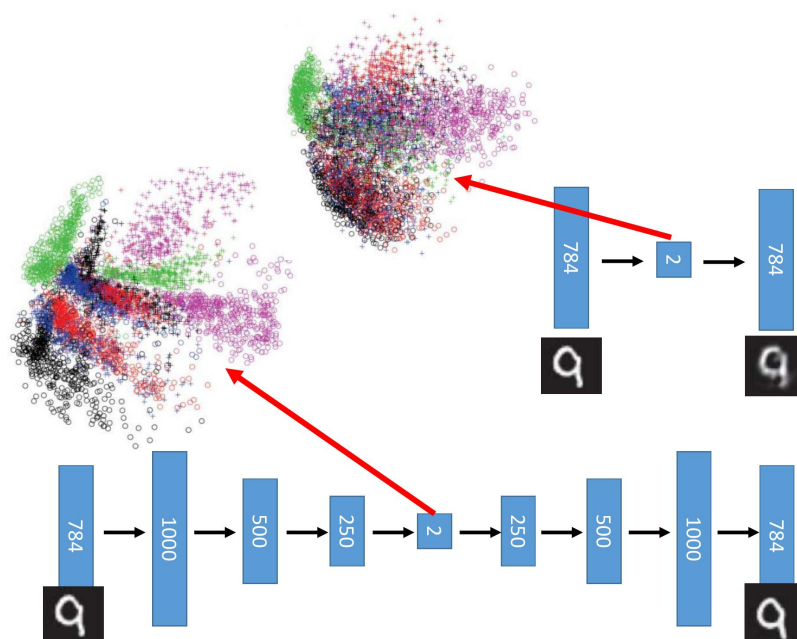
### Deep Auto-encoder



- PCA
  - 784 維降到 30 維
  - 再從 30 維 reconstruct 回 784 維
  - 較模糊的
- deep auto-encoder
  - 784 維，先擴展成 1000 維，再把 1000 維降到 500 維再降到 250 維再降到 30 維
  - 再把 30 維變成 250 維再變成 500 維 1000 維，再解回來 784 維
  - 較清楚

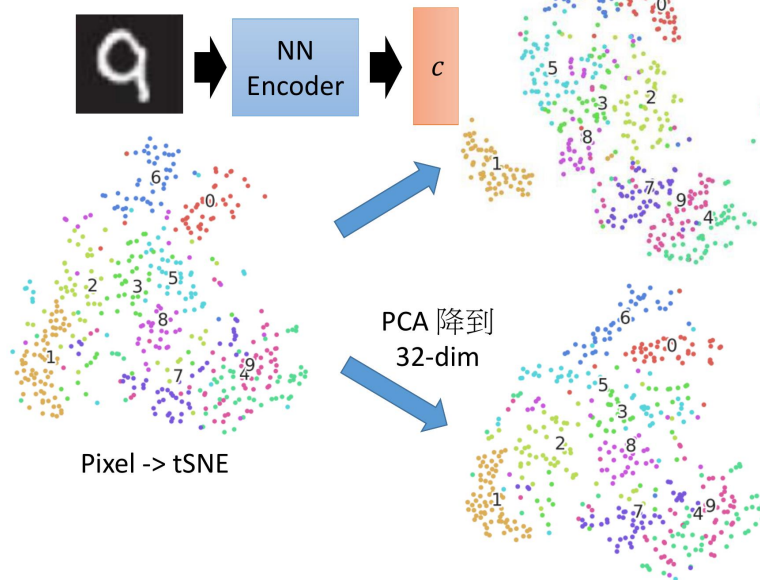
## Compared on 2 dimension

visualize on 2 dimension



- PCA
  - 混在一起
- deep auto-encoder 的話
  - 是分開的
  - 不同的數字會變一群一群

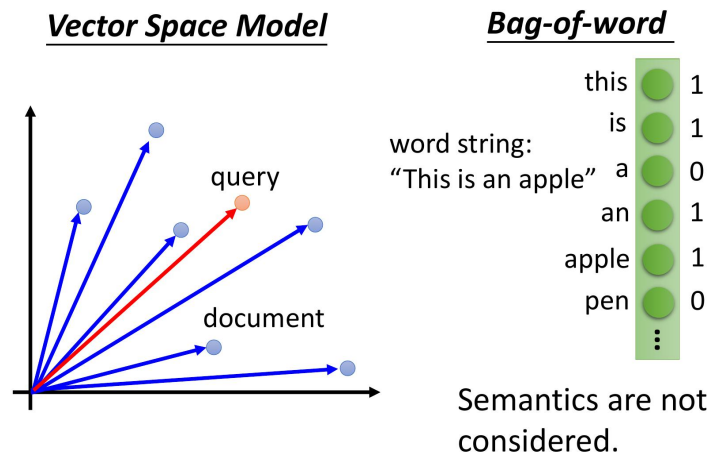
### Deep Auto-encoder - Example



## Use on Text Preprocessing

- 文字搜尋
  - vector space model
    - 文章都表示成空間中的一個 vector
  - 計算輸入的查詢詞彙跟每一篇 document 之間的 inner product 或是 cosine similarity 等等
  - 根據值的大小決定是否 retrieve

# Auto-encoder – Text Retrieval

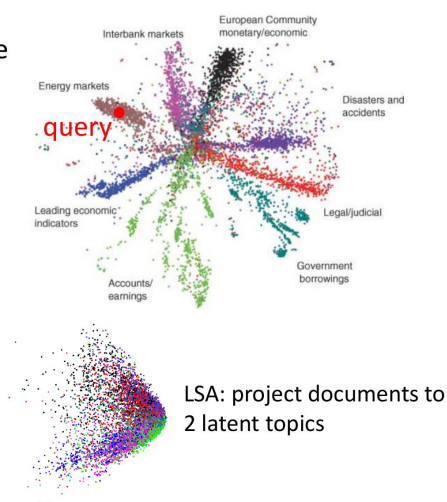
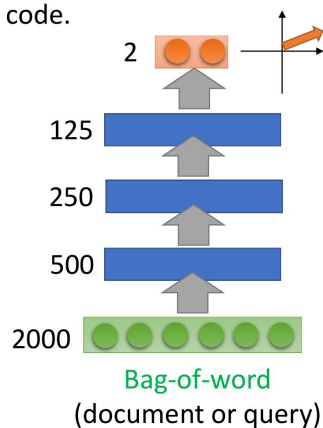


把一個 **document** 變成一個 **vector**

- bag-of-word
  - 最 trivial
  - Term Frequency
  - Inverse Document Frequency
  - TFIDF
  - 沒辦法考慮任何語意相關的東西，每一個詞彙都是 independent

# Auto-encoder – Text Retrieval

The documents talking about the same thing will have close code.

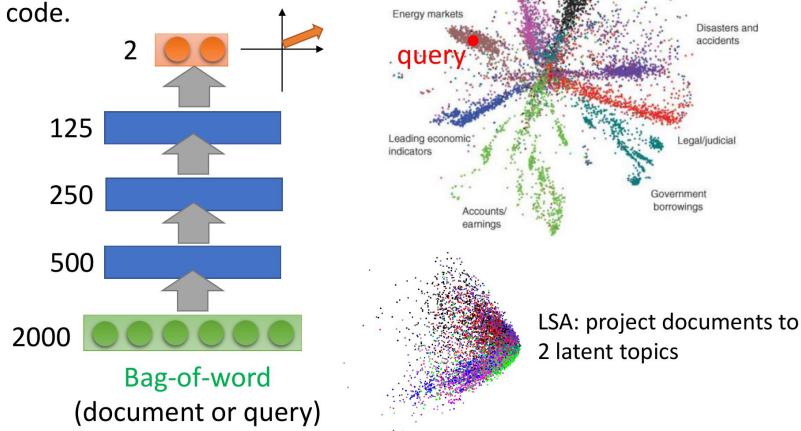


- auto-encoder
  - input: document
  - query: 一段文字

用比較小的 lexicon size，把一個 document 就把它變成一個 vector，再把這個 vector 通過一個 encoder，把他壓成二維，然後 Visualize

# Auto-encoder – Text Retrieval

The documents talking about the same thing will have close code.



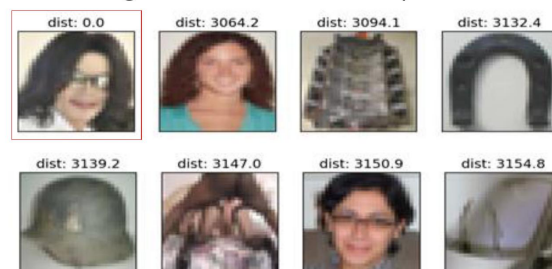
發現同一類的 document，就都集中在一起，散佈像一朵花一樣

要做搜尋的時候

- 輸入一個詞彙、查詢詞
- 把 query 也通過這個 encoder，把他變成一個二維的 vector
- 看query 落在哪邊，就可以知道說這個 query 是哪個 topic

## Use on Image Searching

Retrieved using Euclidean distance in pixel intensity space



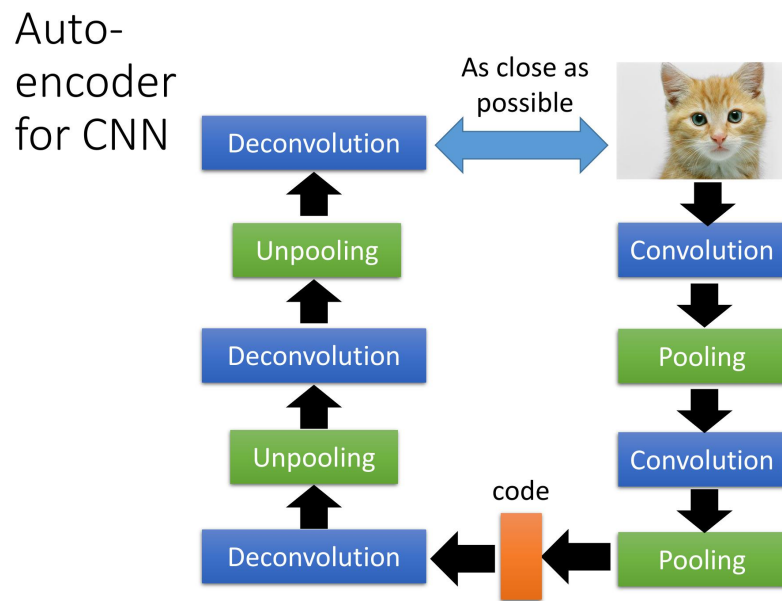
retrieved using 256 codes



- 以圖找圖。
- 最簡單的方法，在 pixel wise 上做比較，但是找不到好的結果的。

## Auto-encoder for CNN

- 應該要用 deep auto-encoder 把每一張 image 變成一個 code，然後在 code 上面再去做搜尋
- 因為這是 unsupervised
- train 這種 auto-encoder 的 data 是永遠不缺的

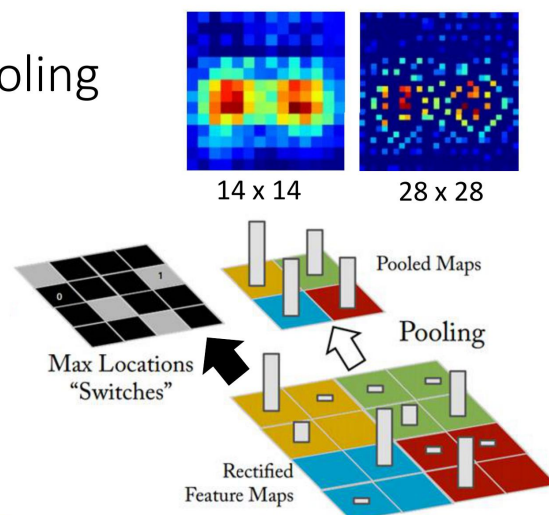


在這種 code 上面算相似度的話，就會得到比較好的結果

如果 encoder 的部分是做 convolution 再做 pooling，convolution 再做 pooling 理論上 decoder 應該就是做跟 encode 相反的事情 本來有 pooling 就做 unpooling，本來有 convolution 就做 deconvolution

## Unpooling

### CNN -Unpooling



Alternative: simply repeat the values

Source of image :  
[https://leonardoraujosantos.gitbooks.io/artificial-intelligence/content/image\\_segmentation.html](https://leonardoraujosantos.gitbooks.io/artificial-intelligence/content/image_segmentation.html)

- 要把原來比較小的 matrix 擴大
- 方法一

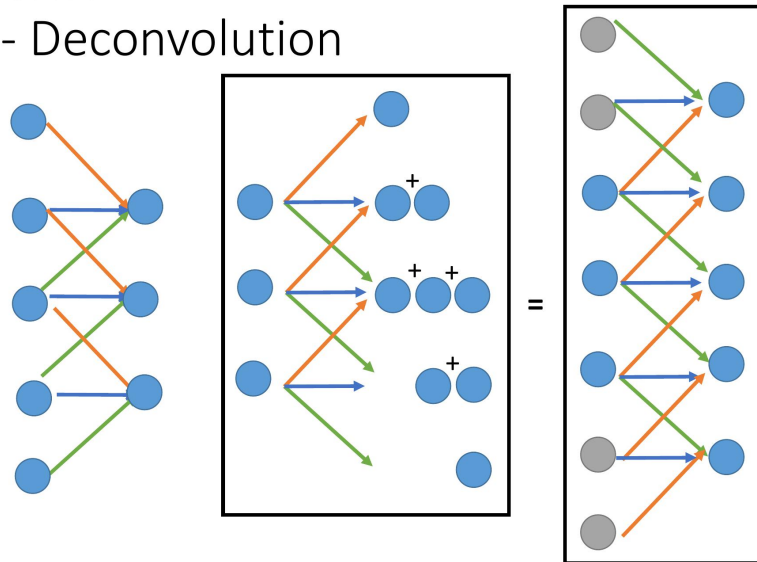
- 記住pooling是從哪裡取值，照樣還原回去
- 沒有取值的部分補0
- 方法二
  - 直接把那個值複製四份，不用去記從哪裡取 (Keras 是用這種方式)

## Deconvolution

CNN

- Deconvolution

Actually, deconvolution is convolution.



- 事實上 deconvolution 就是 convolution
- 不同點是在他們的 weight 是相反
- 做的 operation 一樣也就是 convolution 這件事

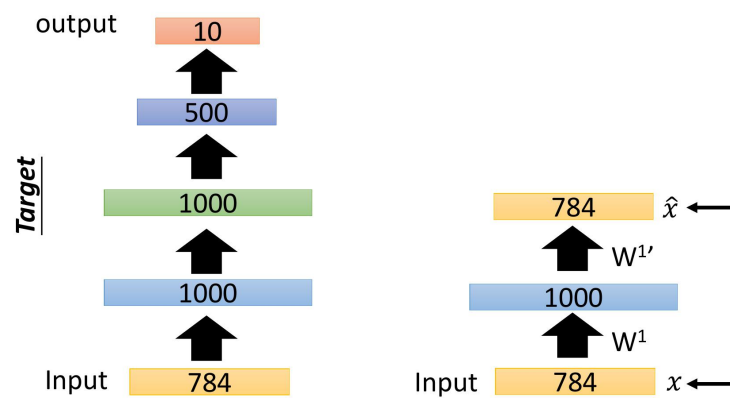
## Using with Pre-training

有時候在煩惱怎麼做參數的 initialization，這種找比較好的 initialization 方法，就叫做 pre-training。那可以用 auto-encoder 來做 pre-training



## Auto-encoder – Pre-training DNN

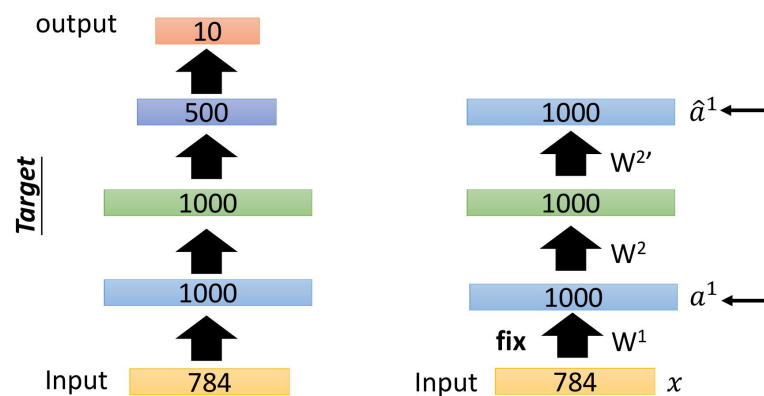
- Greedy Layer-wise Pre-training *again*



先 train 784-1000-784，把 weight( $W_1$ ) 記下來

## Auto-encoder – Pre-training DNN

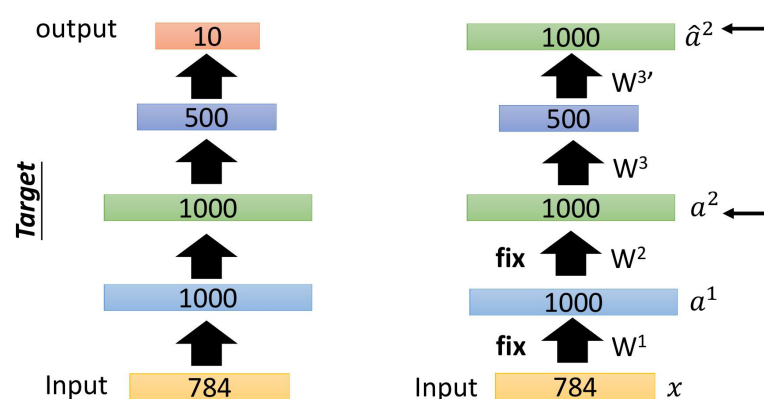
- Greedy Layer-wise Pre-training *again*



再 train 784-1000-1000-1000 784-1000 部分的 weight( $W_1$ ) 用前面的，並 fix 住 一樣把 weight( $W_2$ ) 記下來

## Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*

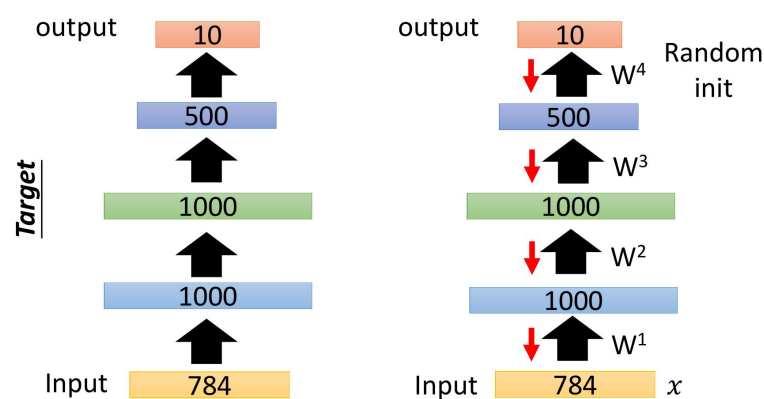


再 train 784-1000-1000-500-1000 784-1000 部分的 weight( $W_1$ ) 以及 1000-1000 部分的 weight( $W_2$ ) 用前面的，並 fix 住 一樣把 weight( $W_3$ ) 記下來

## Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*

Find-tune by  
backpropagation



就可以整個串起來，並把剛剛那些 weight ( $W_1, W_2, W_3$ ) 作為 model 的 initialization，再 random initialize 最後 500 到 100 的 weight，再用 back propagation 去調一遍，我們稱之為 fine tune。

## Generating Outputs

- 那個 decoder 其實是有妙用的，可以拿 decoder 來產生新的 image
- 也就是說我們把 learn 好的 decoder 拿出來，然後給他一個 random 的 input number，output 希望就是一張圖

這件事可以做到嗎，其實這件事做起來相當容易

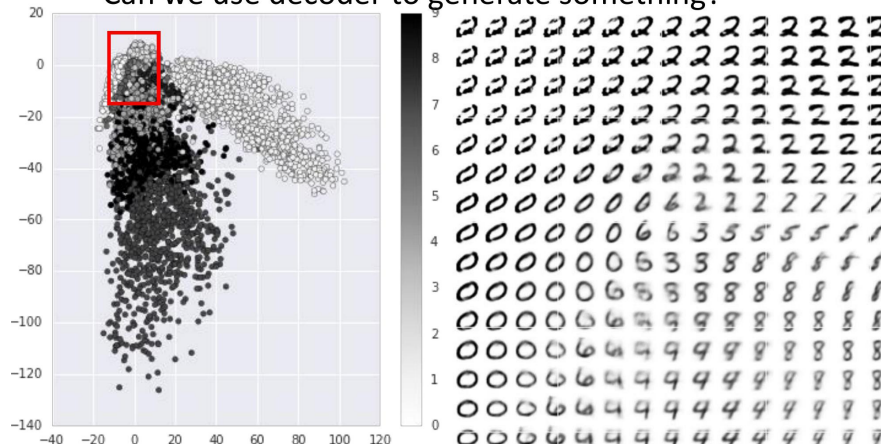
- 在 MNIST dataset 上，把每一張圖，784 維的 image 通過一個 hidden layer 然後 project 到二維

- 再把二維通過一個 hidden layer 解回原來的 image
- 那在 encoder 的部分，那個二維的 vector 畫出來長這樣

Next .....



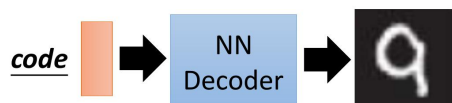
- Can we use decoder to generate something?



- 在紅色這個框框裡面等間隔的去 sample 一個二維的 vector 出來
- 然後把那個二維的 vector 丟到 NN decoder 裡面， output 一個 image 出來
- 可以發現很多有趣的現象
  - 從下到上，感覺是圓圈然後慢慢的就垮了
  - 右下這邊本來是不知道是四還是九，然後變八
  - 再往上然後越來越細，變成 1
  - 最後不知道為什麼變成 2，還蠻有趣的

會發現在這邊感覺比較差，是因為在這邊其實是沒有 image，所以你在 input image 的時候其實不會對到這邊。這個區域的 vector sample 出來，通過 decoder 他解回來不是 image

Next .....



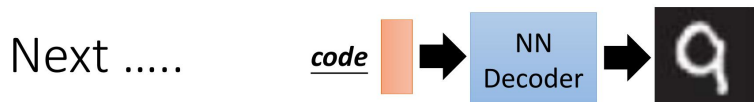
- Can we use decoder to generate something?



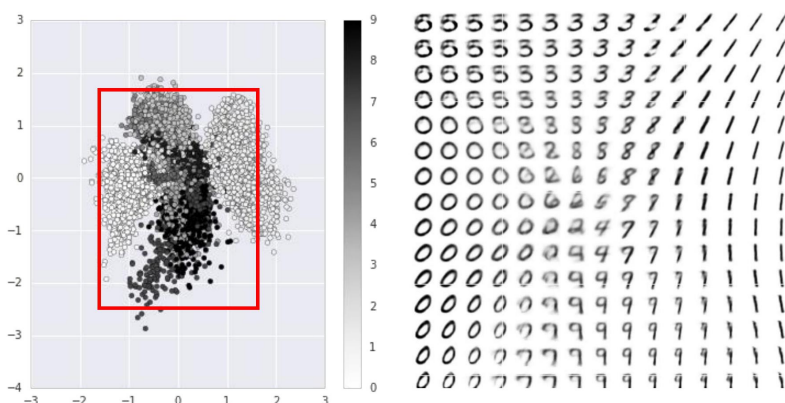
所以要 Sample 到一個好的地方 因為我必須要先觀察一下二維 vector 的分佈，才能知道哪邊是有值的，才知道從那個地方 sample 出來比較有可能是一個 image。

可是這樣你要先分析二維的 code 感覺有點麻煩，有個很簡單的做法就是在你的 code 上面加 regularization。在你的 code 直接加上 L2 的 regularization，讓所有的 code 都比較接近零。接下來就在零附近 sample 就好了。

接下來我就以零為中心，然後等距的在這個紅框內 sample image，sample 出來就這個樣子。



- Can we use decoder to generate something?



從這邊你就可以觀察到很多有趣的現象 會發現說：

- 這個 dimension 是有意義的
  - 從左到右橫軸代表的是有沒有圈圈
  - 縱的呢，本來是正的，然後慢慢就倒過來

所以你可以不只是做 encode，還可以用 code 來畫。這個 image 並不是從原來 image database sample 出來的，他是 machine 自己畫出來的。