- model architecture

layer	Input shape	Filter	Stride	Padding	Output shape
Conv1	(16, 1, 32, 32)	(3, 5, 5)	1*1	no	(16, 3, 28, 28)
ReLU	(16, 3, 28, 28)				(16, 3, 28, 28)
Maxpooling	(16, 3, 28, 28)	(3, 3)	2*2	no	(16, 3, 13, 13)
Flatten	(16, 3, 13, 13)				(16, 3*13*13)
Fully Connected 1	(16, 3*13*13)				(16, 64)
ReLU	(16, 64)				(16, 64)
Fully Connected 2	(16, 64)				(16, 3)
Softmax	(16, 3)				(16, 3)

註 1: Conv1 的 input shape=(16, 1, 32, 32)，16 為 batch size，1 為 channel(將照片灰階模式讀取並做 normalization)
註 2: Conv1 的 filter=(3, 5, 5)，3 為產生三張 feature map，5 為 kernel size
註 3: Maxpooling 的 filter=(3, 3)，意思是在 feature map 中每個 3*3 的 2D 區域內取一個最大值做降維，並且步幅為 2

- loss function

採用多類別交叉熵，公式如下:

$$\mathrm{L} = \frac{-1}{N}\sum_{i=1}^{N}\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

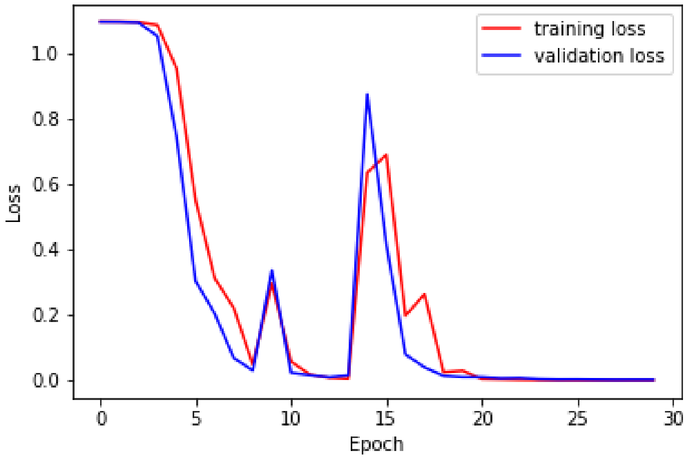
N 為樣本總數，M 為類別數，y_{ic}為 label(one hot vector)，p_{ic}為 model predicted probability

- screenshot of testing result

```
In [1]: runfile('/homes/chen-chun-yu/DL_HW3/main.py', wdir='/homes/chen-chun-yu/DL_HW3')
Namespace(batch_size=16, epoch=30, lr=0.05)
model init...
start training...
EPOCH: 1 , train_loss: 1.0995 , val_loss: 1.0989 , val_acc: 31.75 %
EPOCH: 2 , train_loss: 1.0991 , val_loss: 1.0986 , val_acc: 31.75 %
EPOCH: 3 , train_loss: 1.0983 , val_loss: 1.0968 , val_acc: 31.75 %
EPOCH: 4 , train_loss: 1.0892 , val_loss: 1.0548 , val_acc: 53.06 %
EPOCH: 5 , train_loss: 0.9556 , val_loss: 0.7458 , val_acc: 81.63 %
EPOCH: 6 , train_loss: 0.5532 , val_loss: 0.3047 , val_acc: 84.35 %
EPOCH: 7 , train_loss: 0.3137 , val_loss: 0.205 , val_acc: 95.92 %
EPOCH: 8 , train_loss: 0.222 , val_loss: 0.069 , val_acc: 97.73 %
EPOCH: 9 , train_loss: 0.0483 , val_loss: 0.0305 , val_acc: 99.09 %
EPOCH: 10 , train_loss: 0.2974 , val_loss: 0.3377 , val_acc: 90.25 %
EPOCH: 11 , train_loss: 0.0585 , val_loss: 0.0239 , val_acc: 99.09 %
EPOCH: 12 , train_loss: 0.0178 , val_loss: 0.0161 , val_acc: 99.55 %
EPOCH: 13 , train_loss: 0.0083 , val_loss: 0.0101 , val_acc: 99.55 %
EPOCH: 14 , train_loss: 0.0053 , val_loss: 0.0157 , val_acc: 99.32 %
EPOCH: 15 , train_loss: 0.6366 , val_loss: 0.8771 , val_acc: 62.59 %
EPOCH: 16 , train_loss: 0.692 , val_loss: 0.4174 , val_acc: 82.31 %
EPOCH: 17 , train_loss: 0.1984 , val_loss: 0.0801 , val_acc: 97.28 %
EPOCH: 18 , train_loss: 0.2645 , val_loss: 0.0404 , val_acc: 98.41 %
EPOCH: 19 , train_loss: 0.0255 , val_loss: 0.0141 , val_acc: 99.32 %
EPOCH: 20 , train_loss: 0.03 , val_loss: 0.0105 , val_acc: 99.55 %
EPOCH: 21 , train_loss: 0.0046 , val_loss: 0.0105 , val_acc: 99.32 %
EPOCH: 22 , train_loss: 0.0035 , val_loss: 0.0067 , val_acc: 99.77 %
EPOCH: 23 , train_loss: 0.0024 , val_loss: 0.0077 , val_acc: 99.55 %
EPOCH: 24 , train_loss: 0.0018 , val_loss: 0.0042 , val_acc: 100.0 %
EPOCH: 25 , train_loss: 0.0014 , val_loss: 0.0031 , val_acc: 100.0 %
EPOCH: 26 , train_loss: 0.0012 , val_loss: 0.0032 , val_acc: 100.0 %
EPOCH: 27 , train_loss: 0.0011 , val_loss: 0.0026 , val_acc: 100.0 %
EPOCH: 28 , train_loss: 0.001 , val_loss: 0.0022 , val_acc: 100.0 %
EPOCH: 29 , train_loss: 0.0009 , val_loss: 0.0022 , val_acc: 100.0 %
EPOCH: 30 , train_loss: 0.0007 , val_loss: 0.0022 , val_acc: 100.0 %
select epoch: 28
Testing accuracy: 98.8 %

In [2]:
```

- Plot training loss and validation loss



- Describe the major problem you encountered and how did you deal with it
手刻 CNN 的難度相較手刻 DNN 難上許多，不像 DNN 可以將數組攤平(Flatten)再處理，CNN 要考慮到多維度之間的轉換，光是 input image 的 shape 就有四個維度(batch_size, channel, width, height)，若 input image 的 channel 為 3 (RGB)，那情況會更加複雜，我的作法是將其以灰階模式讀取(一層 channel)再處理。另外就是模型參數的梯度更新，因為層數多，需要非常細心地做偏微分與連鎖率運算。