# CIFAR 10 Classification by CNN

- From 79% to 90% Accuracy

醫工系107 吳旻昇

# 優化 CNN 的方法

- 改善 Gradient Vanishing 問題：
  1. Xavier Initialization → He Initialization
  2. Batch Normalization
  3. Selu (Self-Normalizing ReLU, SNN)
- 讓 Dropout 比例不一：Differential Dropout
- 隨機打亂資料：Random Shuffling
- 考慮模型複雜度的懲罰值：L2 Regularization
- Learning rate decay
- NIN
- Data Augmentation

# 先講結論

- 使用 Xavier Initialization + Selu 讓 Accuracy 提升至 79%

| method | architecture | epochs | batch_size | keep_probability | conv_num_outputs | conv_ksize | conv_strides | pool_ksize | pool_strides | fully_conn_num_outputs | testing accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Orinial | 1層CNN+ 1層FC | 100 | 512 | 0.3 | 20 | 3 x 3 | 1 x 1 | 8 x 8 | 1 x 1 | 500 | 71.76% |
| Xavier+BN | 1層CNN+ 1層FC | 100 | 512 | 0.3 | 20 | 3 x 3 | 1 x 1 | 8 x 8 | 1 x 1 | 500 | 74.57% |
| Selu | 1層CNN+ 1層FC | 100 | 512 | 0.3 | 20 | 3 x 3 | 1 x 1 | 8 x 8 | 1 x 1 | 500 | 74.62% |
| Xavier+Selu | 1層CNN+ 1層FC | 100 | 128 | 0.3 | 24 | 3 x 3 | 1 x 1 | 2 x 2 | 1 x 1 | 512 | 74.72% |
| Xavier+BN with Multilayer | 6層CNN+ 2層FC (每2層1個 dropout) | 100 | 128 | 0.3 | 24 | 3 x 3 | 1 x 1 | 2 x 2 | 1 x 1 | 512 | 76.53% |
| Xavier+Selu with Multilayer | 6層CNN+ 2層FC (每2層1個 dropout) | 100 | 128 | 0.3 | 24 | 3 x 3 | 1 x 1 | 2 x 2 | 1 x 1 | 512 | 79.31% |

Constricted by OOM Problem

# Xavier Initialization

- 讓權重的初始值之高斯分布的變異數與 neuron 個數有關。

$$Var(W) = \frac{1}{n_{in}}$$

- 假設有一 linear neuron layer，希望讓 input 與 output 的變異數一樣。

$$Y = W_1X_1 + W_1X_1 + \cdots + W_nX_n$$

$$\mathrm{Var}(W_iX_i) = E[X_i]^2\mathrm{Var}(W_i) + E[W_i]^2\mathrm{Var}(X_i) + \mathrm{Var}(W_i)\mathrm{Var}(i_i)$$

$$\mathrm{Var}(W_iX_i) = \mathrm{Var}(W_i)\mathrm{Var}(X_i)$$

$$\mathrm{Var}(Y) = \mathrm{Var}(W_1X_1 + W_2X_2 + \cdots + W_nX_n) = n\mathrm{Var}(W_i)\mathrm{Var}(X_i)$$

$$\mathrm{Var}(W_i) = \frac{1}{n} = \frac{1}{n_{\mathrm{in}}}$$

- 對於 non-linear ， He 等人提出 He Initialization。

$$Var(W) = \frac{2}{n_{in}}$$

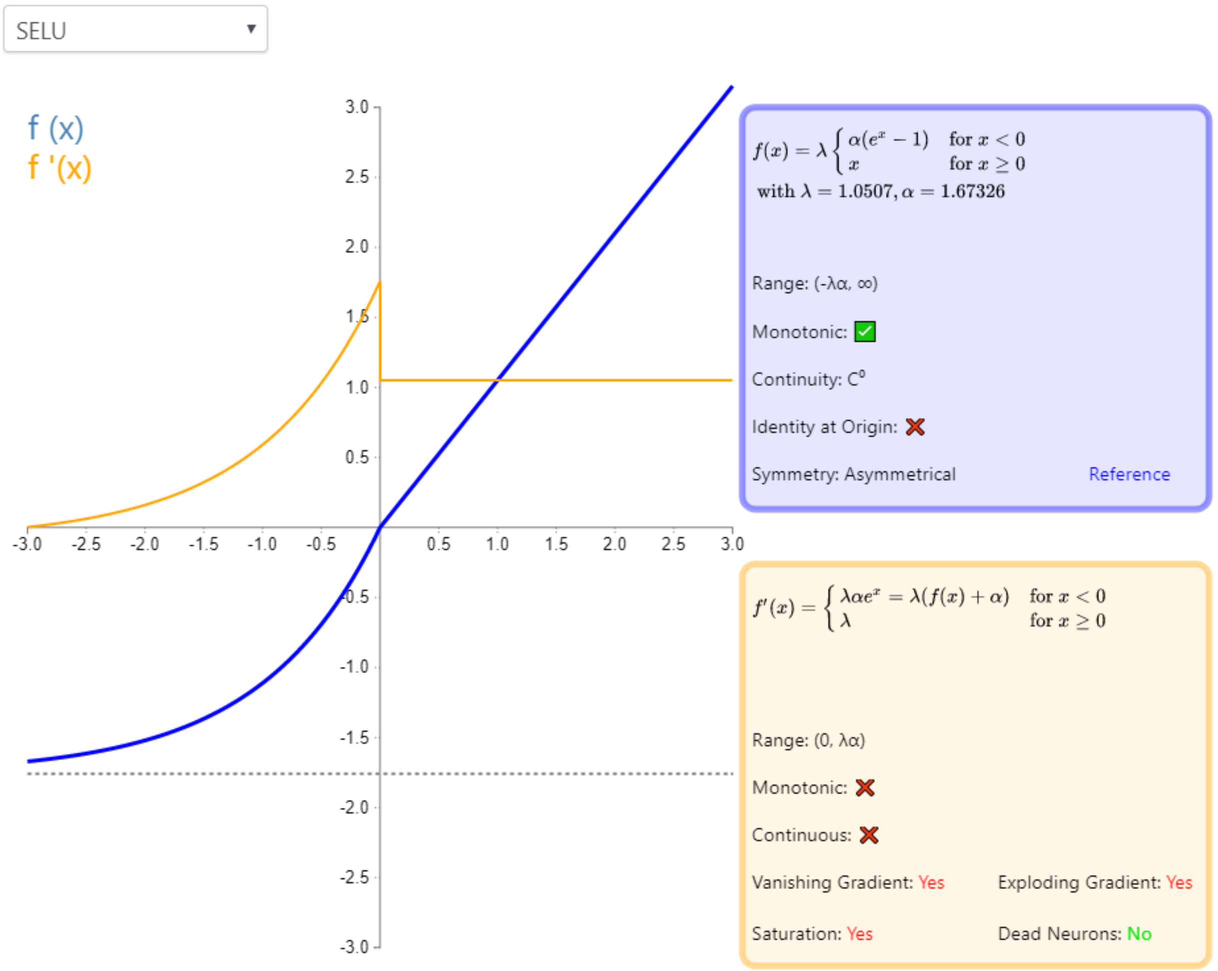
# Selu (Self-Normalizing ReLU , SNN)

- Günter Klambaue et al., Andreas May. Self-Normalizing Neural Networks (2017)

$$Selu(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leq 0 \end{cases}$$

$$\begin{cases} \lambda = 1.0507009873554804934193349852946 \\ \alpha = 1.6732632423543772848170429916717 \end{cases}$$

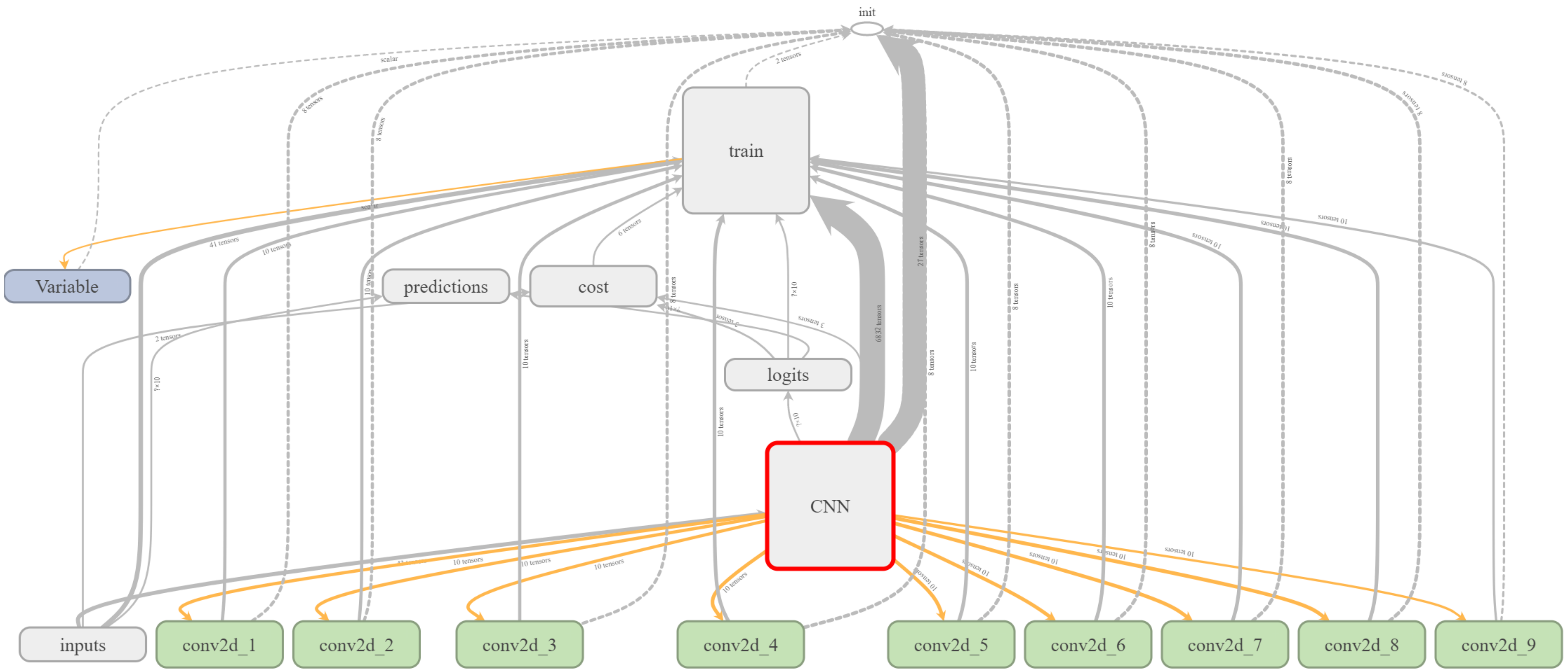
- 自動讓輸出值正規化。




**Scaled Exponential Linear Unit** (**SELU**) is simply a variation on the Exponetial Linear Unit (ELU) activation function, where λ and α have been fixed (to 1.0507 and 1.67326, respectively). The reasoning behind these values (zero mean/unit variance) forms the basis of Self-Normalising Neural Networks (SNNs- see the Reference link in the blue box for more information).

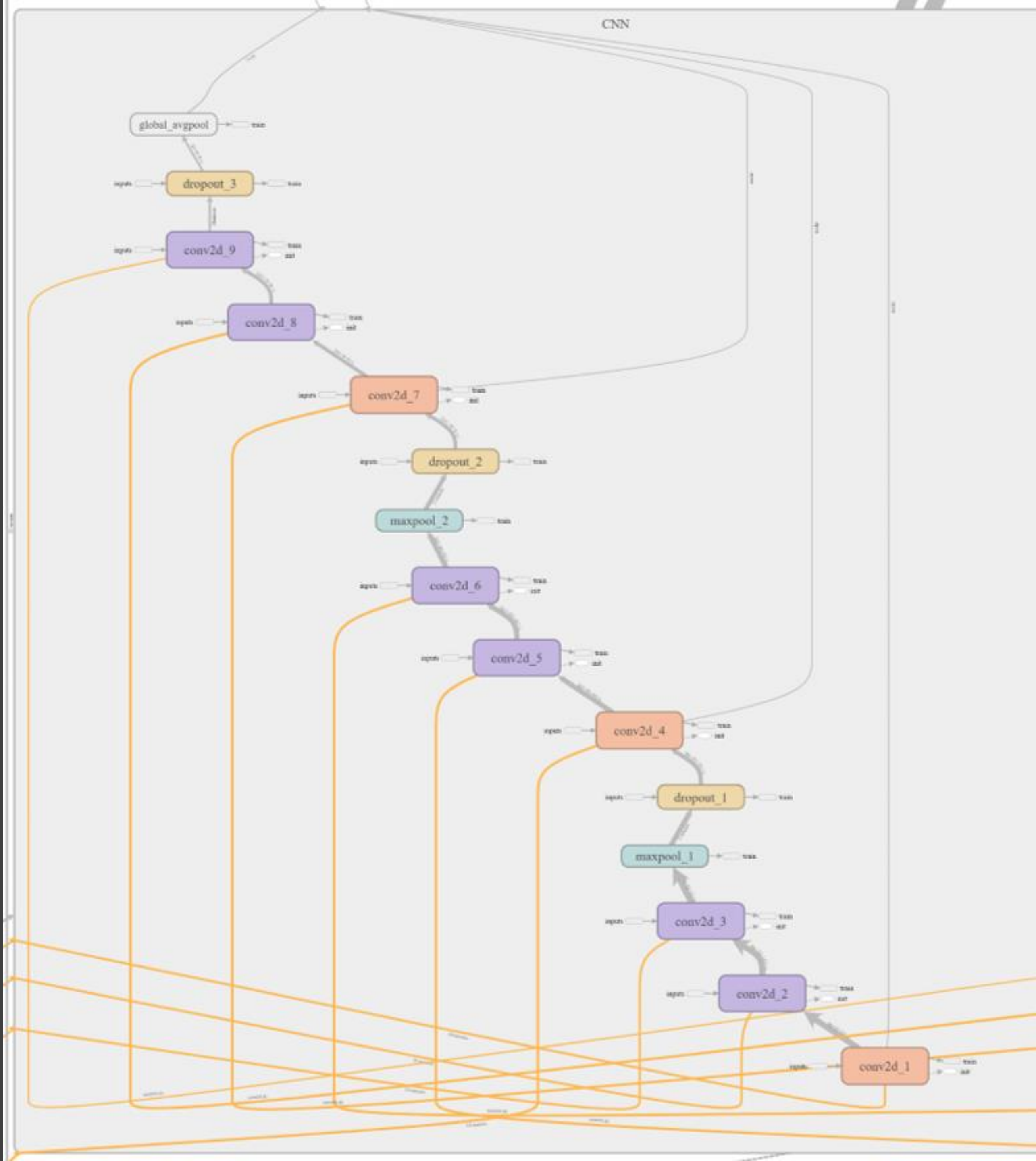
[Different Activation Functions Displaying]

# Final CNN Architecture



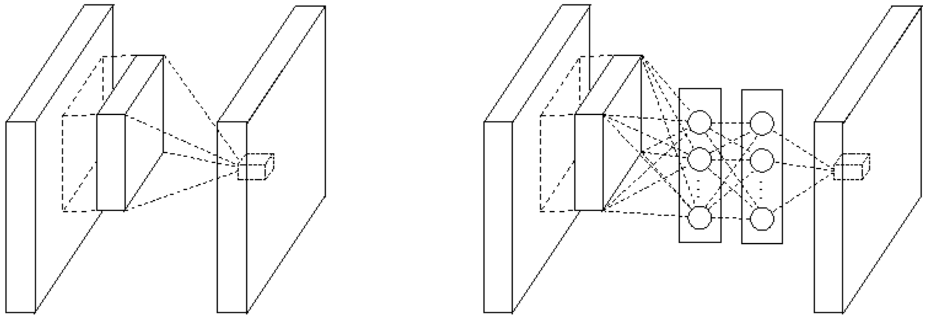
# CNN Layers
## - NIN (Network in Network)

| 9 weight layers | |
|---|---|
| conv2d_1 | 3x3 conv 92 ReLU with l2 regularizer |
| conv2d_2 | 3x3 conv 92 ReLU |
| conv2d_3 | 3x3 conv 92 ReLU |
| maxpool_1 | maxpool with stride = 2 |
| dropout_1 | dropout with keep-prob = 0.5 |
| conv2d_4 | 3x3 conv 192 ReLU with l2 regularizer |
| conv2d_5 | 3x3 conv 192 ReLU |
| conv2d_6 | 3x3 conv 192 ReLU |
| maxpool_2 | maxpool with stride = 2 |
| dropout_2 | dropout with keep-prob = 0.5 |
| conv2d_7 | 3x3 conv 192 ReLU with l2 regularizer |
| conv2d_8 | 1x1 conv 192 ReLU |
| conv2d_9 | 1x1 conv 10 ReLU |
| global_avgpool | global average pooling |

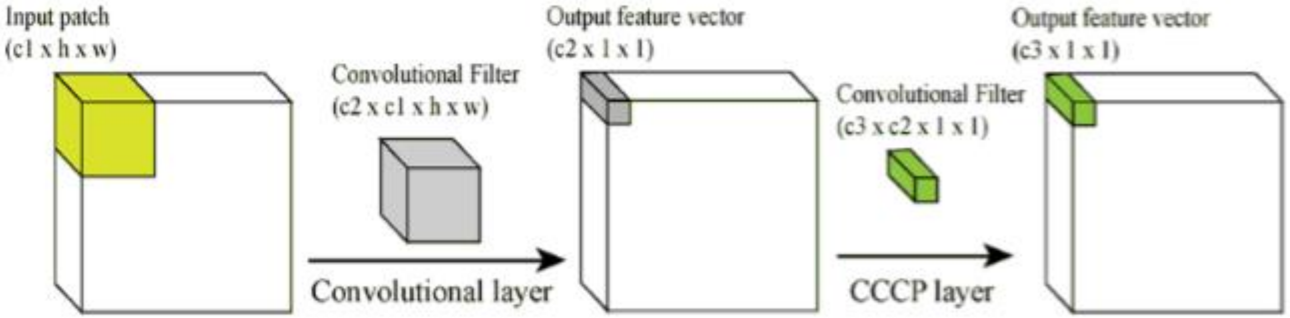


# Mlpconv (multilayer perceptron convolution)

- 在 conv layers 中間加上多層的 fully connected layers。
- 實作上使用 1x1 conv  (又稱 cascaded cross-channel pooling, CCCP)。
- 提高 conv layer 的非線性感知能力。

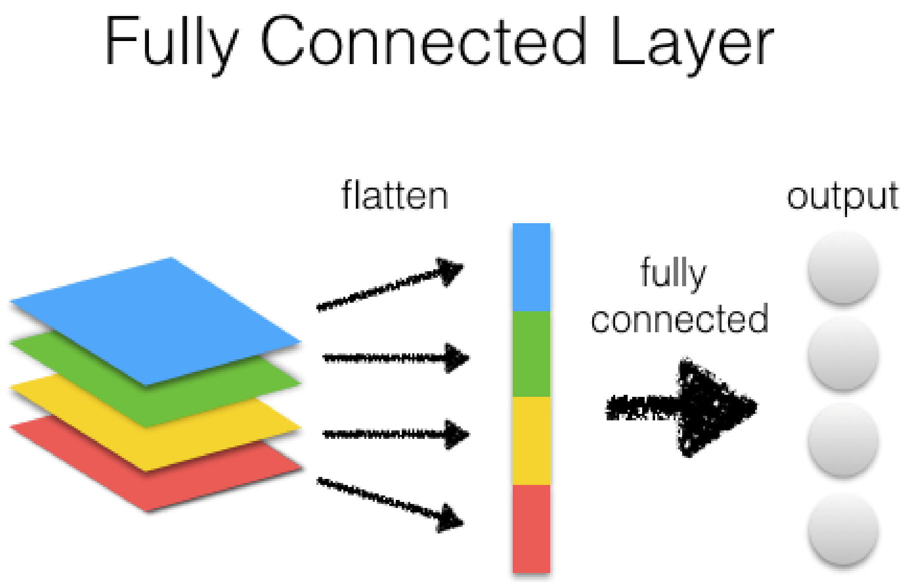


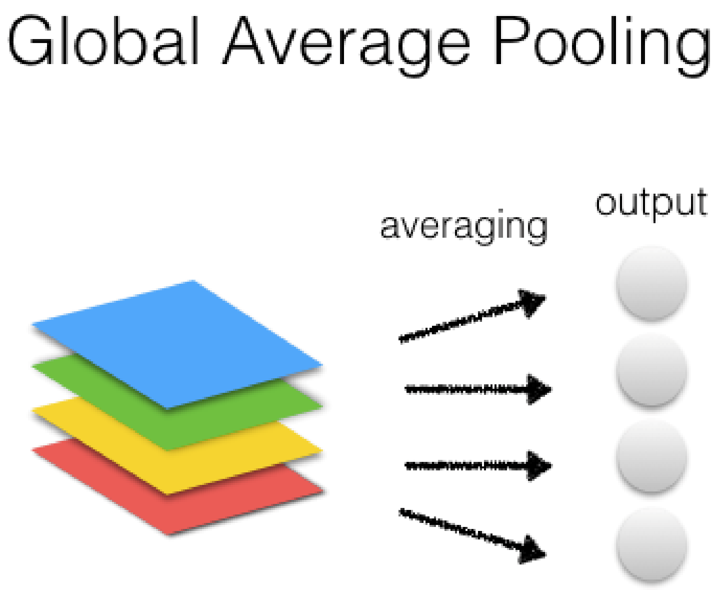
(a) Linear convolution layer (b) Mlpconv layer

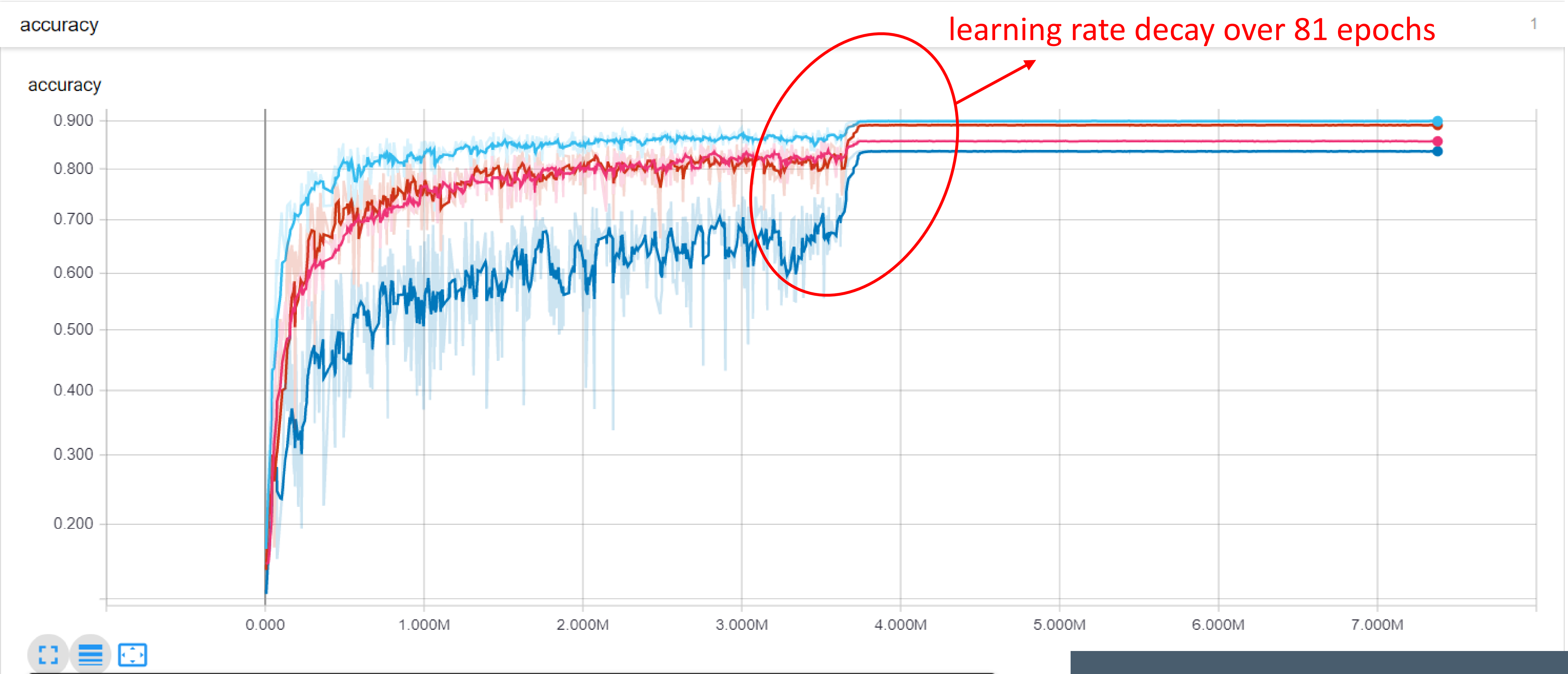


# Global Average Pooling

- 對整個 feature map 取平均值，直接作為output layer的結果。
- 取代 fully connected layers ，大幅減少 parameters 。



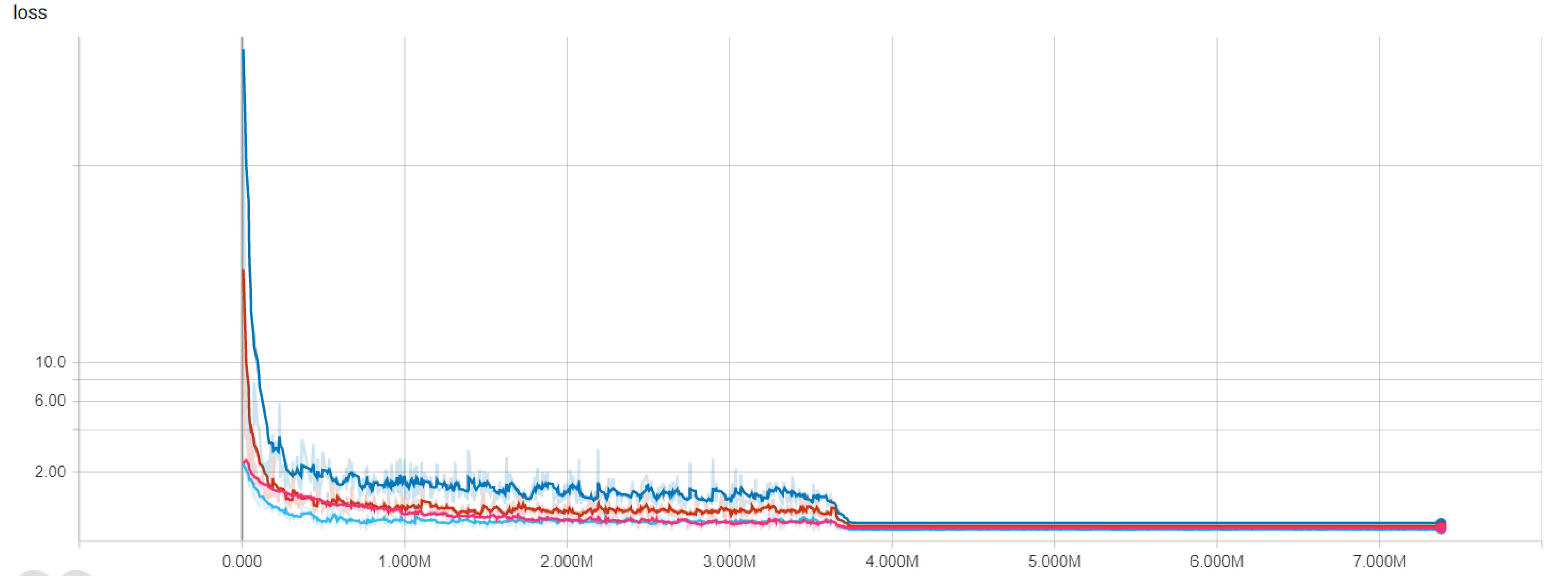






| | Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|---|
| | log\valid, learning_rate=0.00010, batch_size=128 | 0.8569 | 0.8572 | 7.379M | Mon Apr 23, 22:30:33 | 2h 35m 18s |
| | log\valid, learning_rate=0.00100, batch_size=128 | 0.8993 | 0.8990 | 7.379M | Mon Apr 23, 19:55:03 | 2h 34m 10s |
| | log\valid, learning_rate=0.01000, batch_size=128 | 0.8910 | 0.8906 | 7.379M | Mon Apr 23, 17:20:41 | 2h 35m 22s |
| | log\valid, learning_rate=0.10000, batch_size=128 | 0.8362 | 0.8358 | 7.379M | Mon Apr 23, 14:45:06 | 2h 33m 32s |

# Choosing Learning Rate

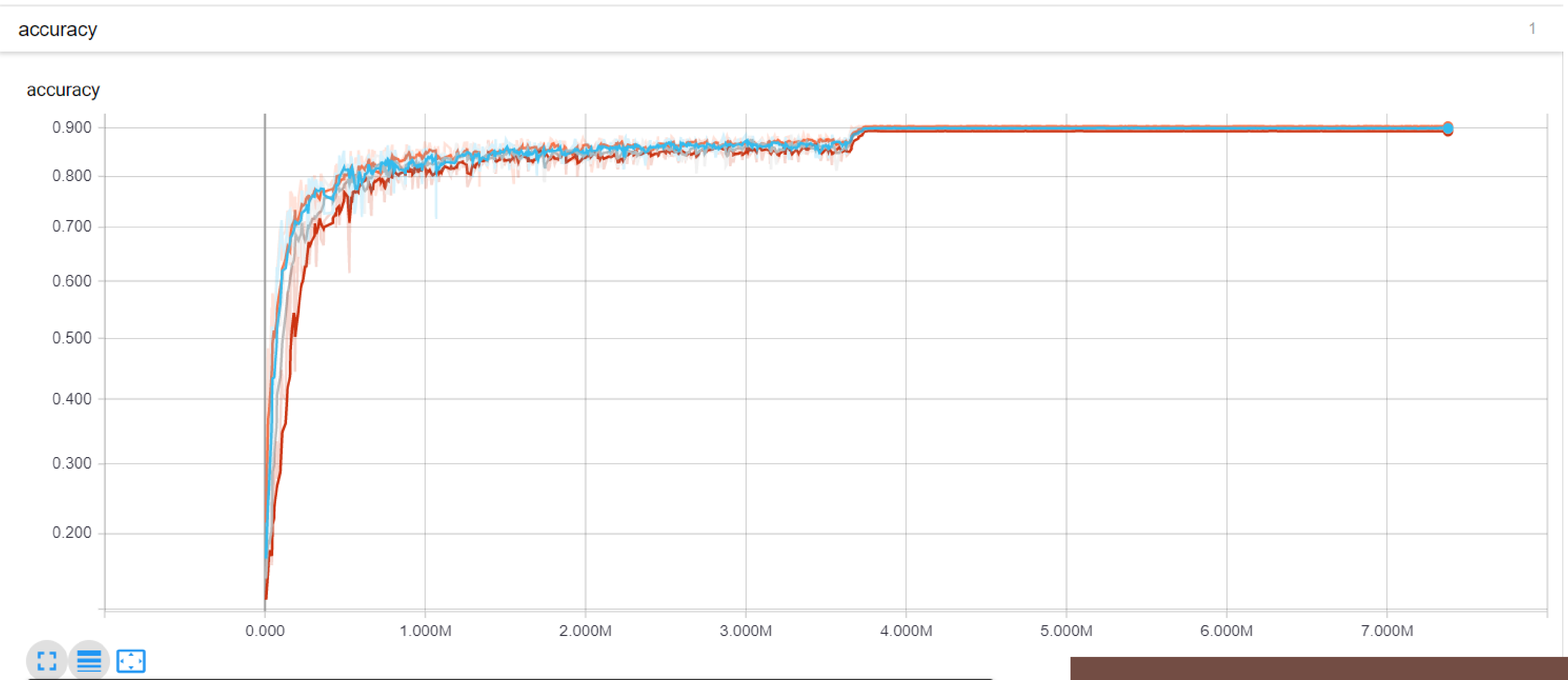
loss 1




| Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|
| log\valid, learning_rate=0.00010, batch_size=128 | 0.5069 | 0.5071 | 7.379M | Mon Apr 23, 22:30:33 | 2h 35m 18s |
| log\valid, learning_rate=0.00100, batch_size=128 | 0.4865 | 0.4860 | 7.379M | Mon Apr 23, 19:55:03 | 2h 34m 10s |
| log\valid, learning_rate=0.01000, batch_size=128 | 0.5400 | 0.5397 | 7.379M | Mon Apr 23, 17:20:41 | 2h 35m 22s |
| log\valid, learning_rate=0.10000, batch_size=128 | 0.6011 | 0.6012 | 7.379M | Mon Apr 23, 14:45:06 | 2h 33m 32s |

# Choosing Learning Rate

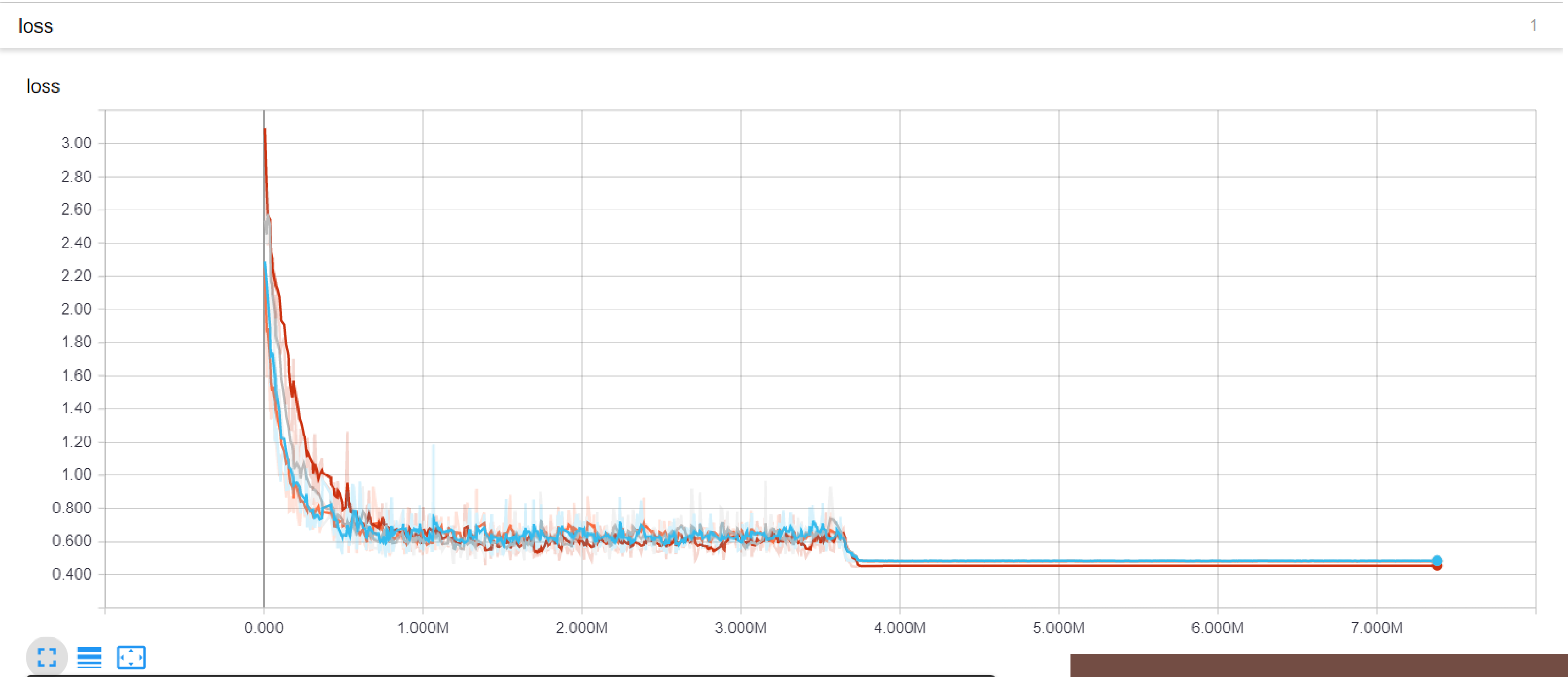
# Choosing Batch Size



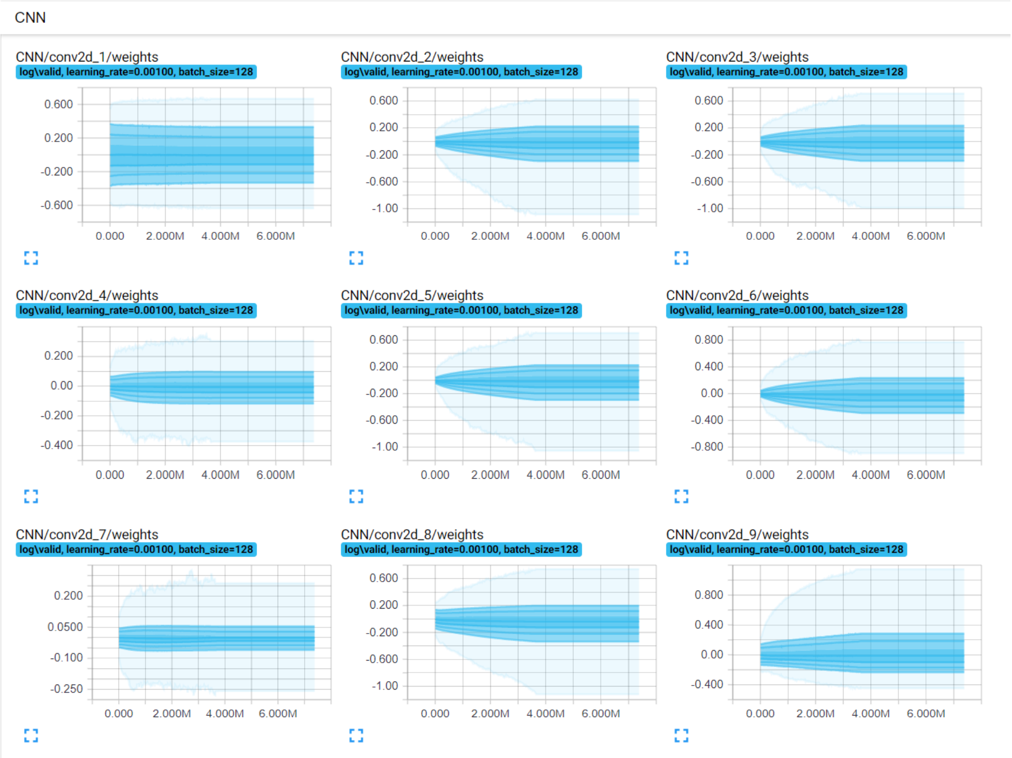

| Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|
| log\valid, learning_rate=0.00100, batch_size=128 | 0.8993 | 0.8990 | 7.379M | Mon Apr 23, 19:55:03 | 2h 34m 10s |
| log\valid, learning_rate=0.00100, batch_size=256 | 0.8984 | 0.8982 | 7.379M | Mon Apr 23, 09:38:52 | 2h 31m 55s |
| log\valid, learning_rate=0.00100, batch_size=512 | 0.8931 | 0.8932 | 7.379M | Sun Apr 22, 23:34:14 | 2h 29m 19s |
| log\valid, learning_rate=0.00100, batch_size=64 | 0.9032 | 0.9032 | 7.379M | Tue Apr 24, 06:38:35 | 2h 42m 45s |

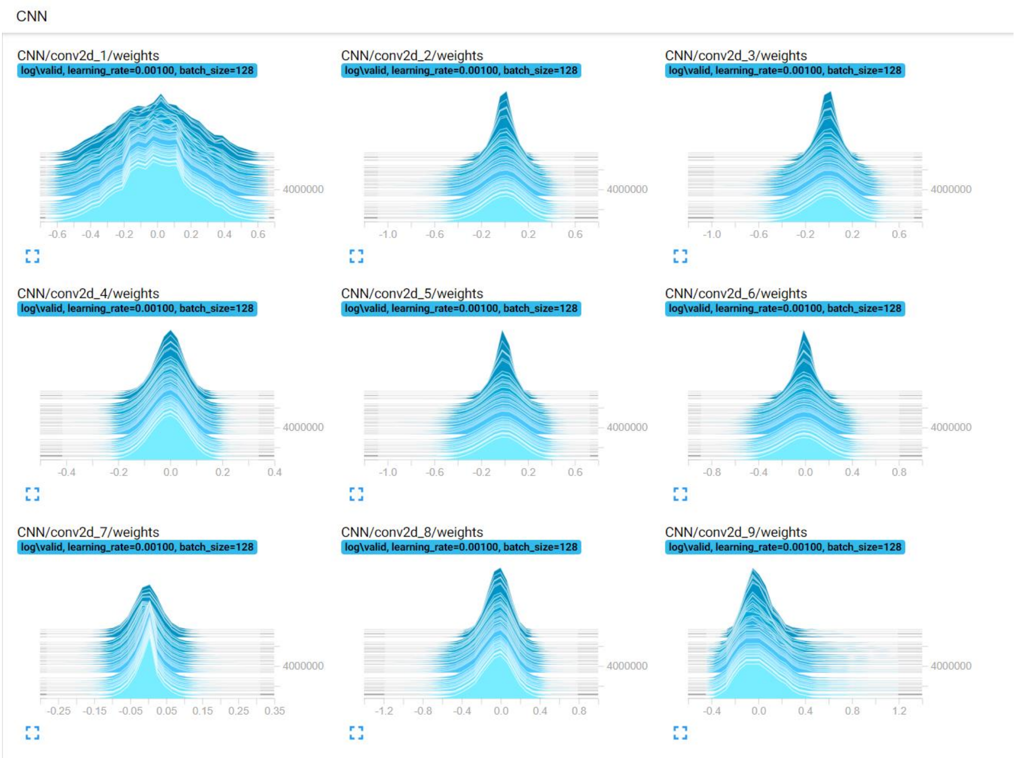
los

# Choosing Batch Size




| | Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|---|
| | log\valid, learning_rate=0.00100, batch_size=128 | 0.4865 | 0.4860 | 7.379M | Mon Apr 23, 19:55:03 | 2h 34m 10s |
| | log\valid, learning_rate=0.00100, batch_size=256 | 0.4860 | 0.4866 | 7.379M | Mon Apr 23, 09:38:52 | 2h 31m 55s |
| | log\valid, learning_rate=0.00100, batch_size=512 | 0.4552 | 0.4556 | 7.379M | Sun Apr 22, 23:34:14 | 2h 29m 19s |
| | log\valid, learning_rate=0.00100, batch_size=64 | 0.4842 | 0.4839 | 7.379M | Tue Apr 24, 06:38:35 | 2h 42m 45s |


CNN
CNN/conv2d_1/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_2/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_3/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_4/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_5/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_6/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_7/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_8/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_9/weights
log\valid, learning_rate=0.00100, batch_size=128


CNN
CNN/conv2d_1/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_2/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_3/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_4/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_5/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_6/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_7/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_8/weights
log\valid, learning_rate=0.00100, batch_size=128
CNN/conv2d_9/weights
log\valid, learning_rate=0.00100, batch_size=128

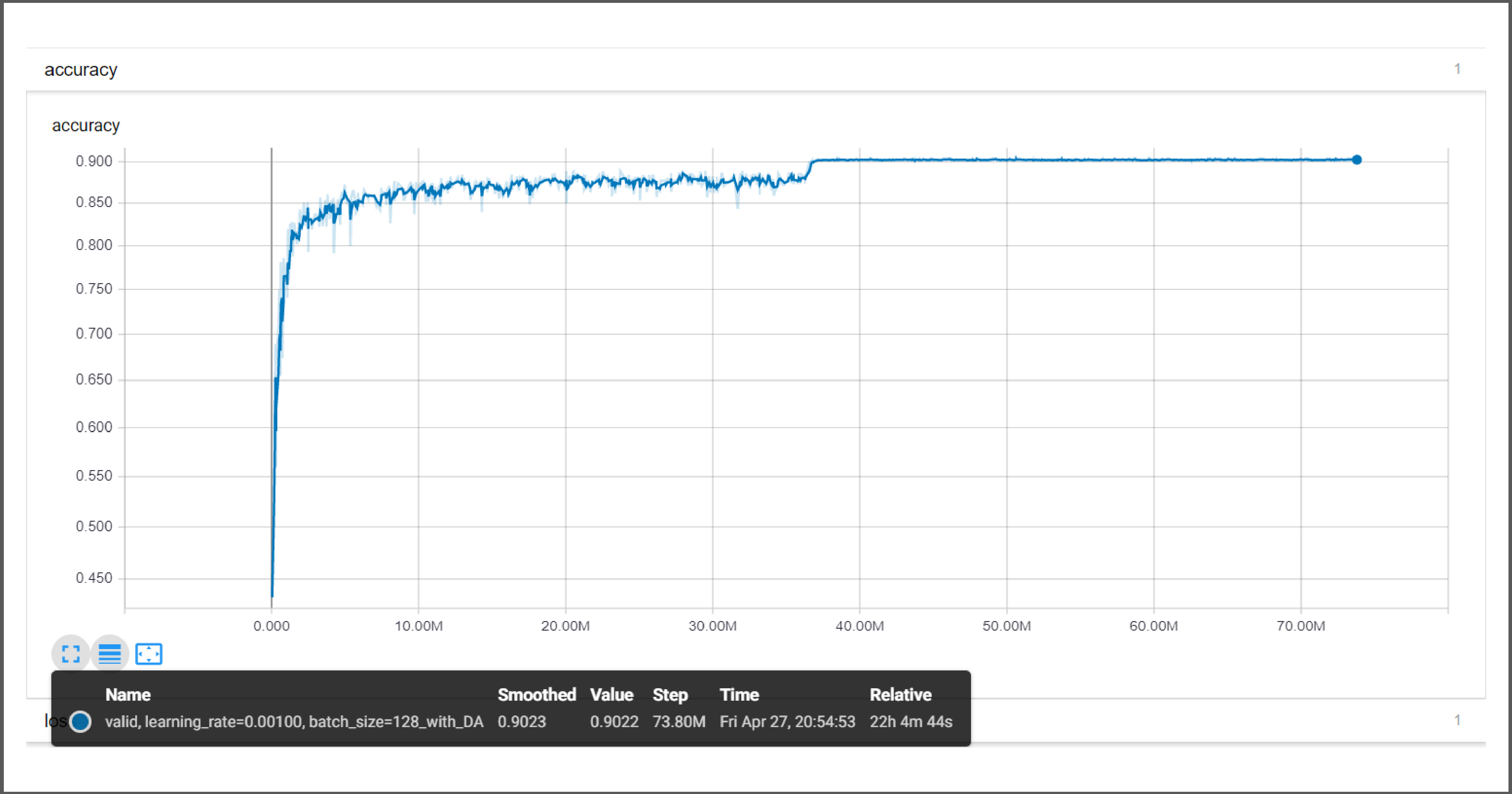
# How to achieve 90% accuracy?

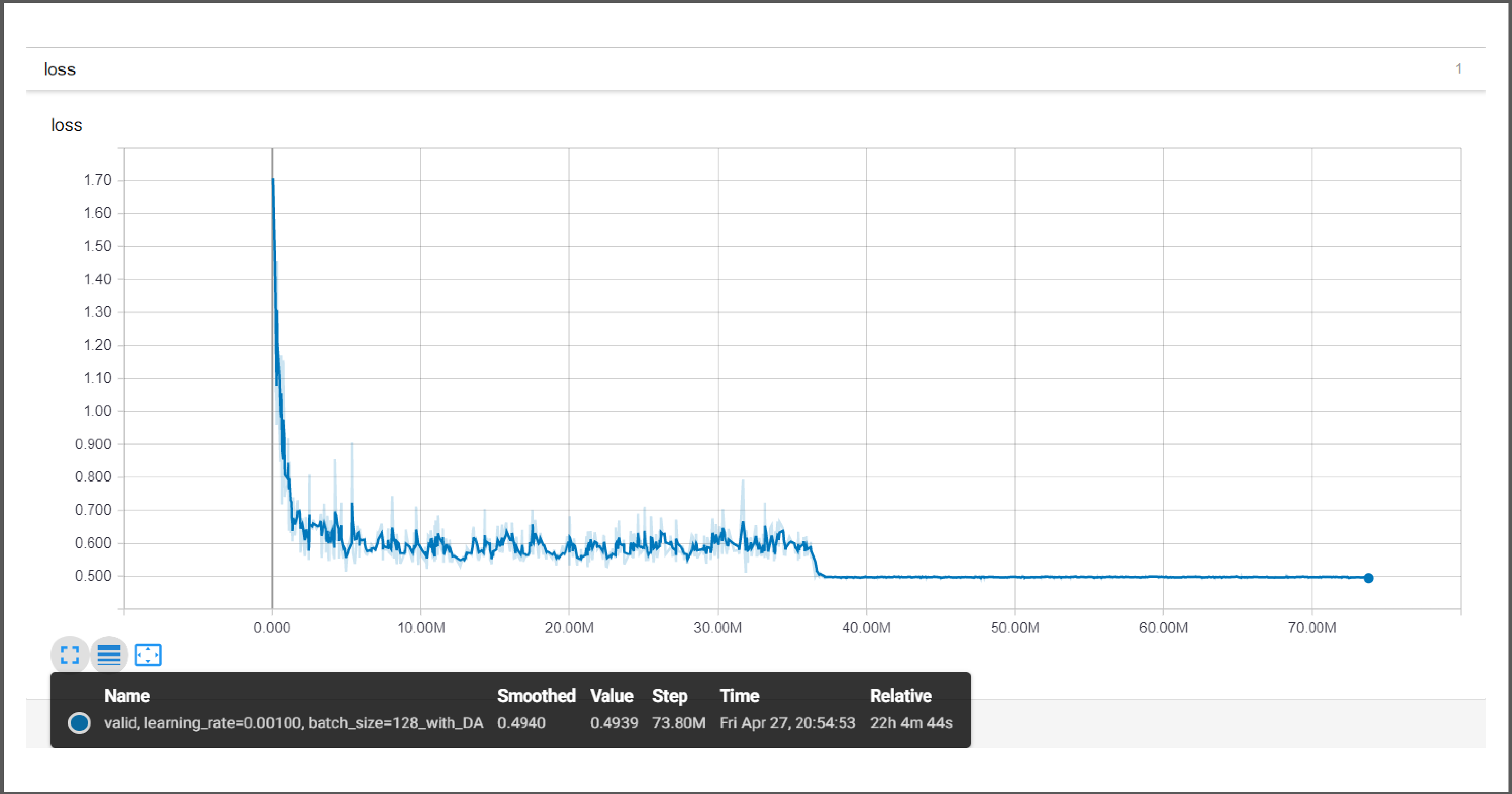
# Data Augmentation

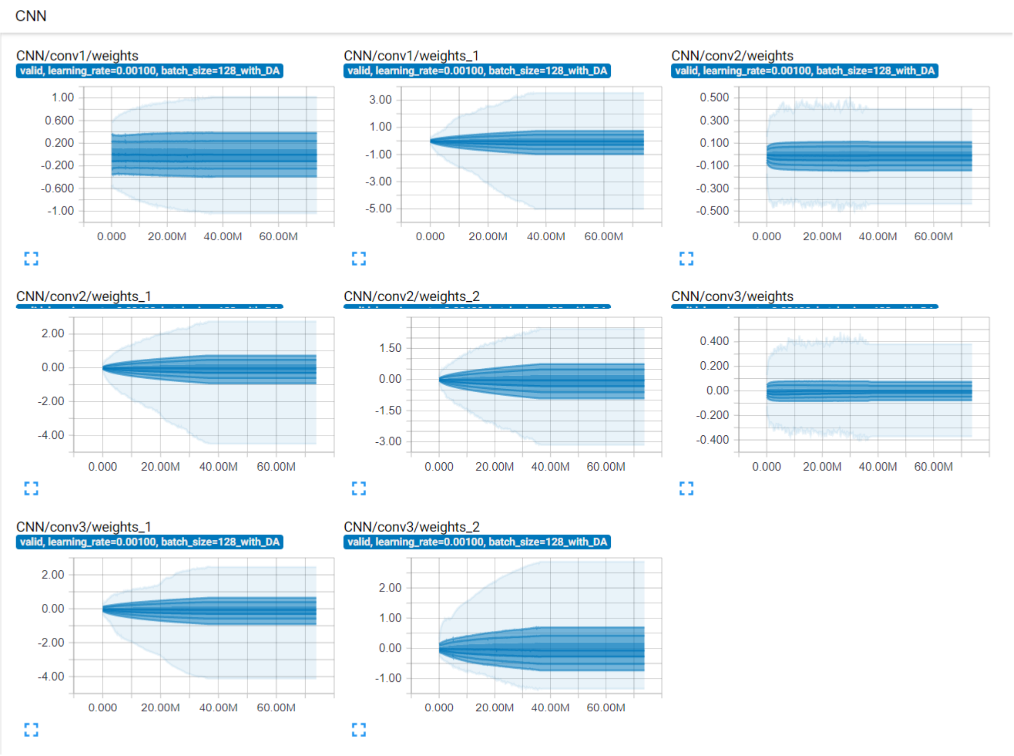
| Kind | | Method | Percentage of images |
|---|---|---|---|
| Horizontal flip | | Random | 50% |
| Crop from each side | | Random | 0 ~ 10% |
| Gaussian blur | | Random sigma 0 ~ 0.5 | 50% |
| Contrast | | 0.75 ~ 1.5 | 100% |
| Add gaussian noise | | per pixel | 50% |
| | | per pixel and per channel | 50% |
| Multiply | | 0.8 ~ 1.2 | 20% |
| Affine transformations | scale | 0.8 ~ 1.2x | 100% |
| | translate | x = -0.2 ~ 0.2<br>y = -0.2 ~ 0.2 | |
| | rotate | -25 ~ 25 deg | |
| | shear | -8 ~ 8 deg | |

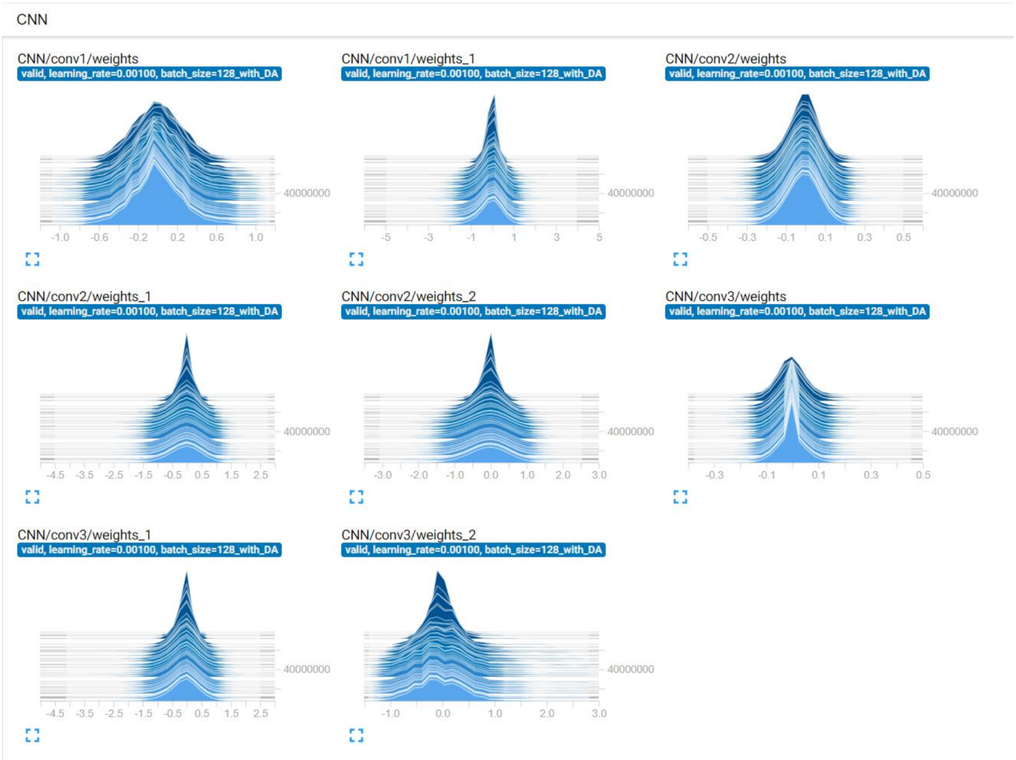
https://github.com/aleju/imgaug

```
aug batch 1 shape:  (81000, 32, 32, 3)
aug batch 2 shape:  (81000, 32, 32, 3)
aug batch 3 shape:  (81000, 32, 32, 3)
aug batch 4 shape:  (81000, 32, 32, 3)
aug batch 5 shape:  (81000, 32, 32, 3)
```


accuracy
1
accuracy
0.900
0.850
0.800
0.750
0.700
0.650
0.600
0.550
0.500
0.450
0.000
10.00M
20.00M
30.00M
40.00M
50.00M
60.00M
70.00M
Name
Smoothed
Value
Step
Time
Relative
valid, learning_rate=0.00100, batch_size=128_with_DA
0.9023
0.9022
73.80M
Fri Apr 27, 20:54:53
22h 4m 44s
los
1


loss
1
loss
1.70
1.60
1.50
1.40
1.30
1.20
1.10
1.00
0.900
0.800
0.700
0.600
0.500
0.000
10.00M
20.00M
30.00M
40.00M
50.00M
60.00M
70.00M
Name Smoothed Value Step Time Relative
valid, learning_rate=0.00100, batch_size=128_with_DA 0.4940 0.4939 73.80M Fri Apr 27, 20:54:53 22h 4m 44s

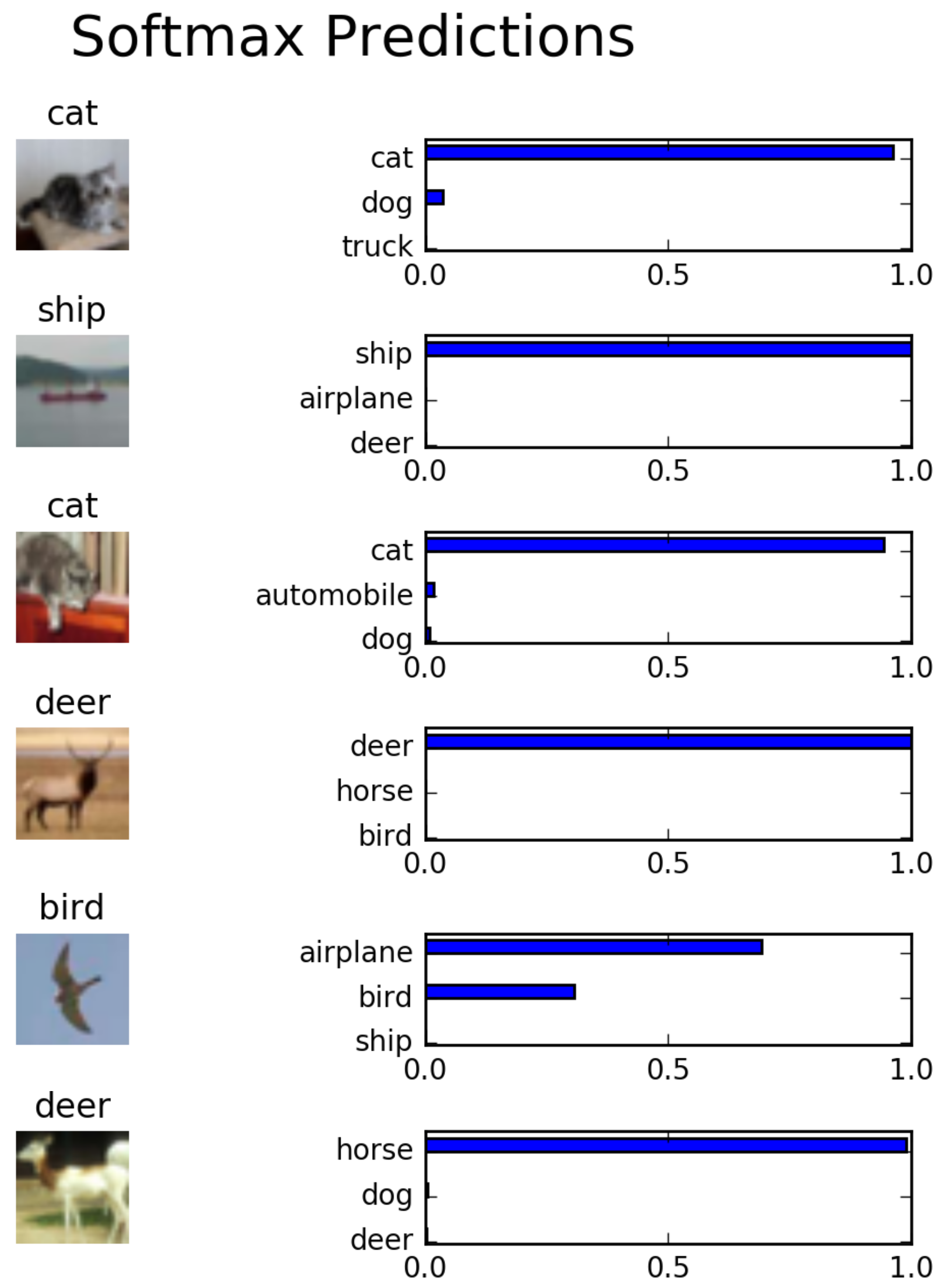

CNN
CNN/conv1/weights
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv1/weights_1
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv2/weights
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv2/weights_1
CNN/conv2/weights_2
CNN/conv3/weights
CNN/conv3/weights_1
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv3/weights_2
valid, learning_rate=0.00100, batch_size=128_with_DA


CNN
CNN/conv1/weights
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv1/weights_1
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv2/weights
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv2/weights_1
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv2/weights_2
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv3/weights
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv3/weights_1
valid, learning_rate=0.00100, batch_size=128_with_DA
CNN/conv3/weights_2
valid, learning_rate=0.00100, batch_size=128_with_DA

# Final Testing Accuracy

- Batch size : 128
- Initial learning rate : 0.001
- Epochs : 164

```
Testing Accuracy: 0.8981408227848101
```



# Reference


- 《Network In Network》
  https://arxiv.org/pdf/1312.4400.pdf
- 《Striving For Simplicity：The All Convolutional Net》
  https://arxiv.org/pdf/1412.6806.pdf
- NIN
  https://github.com/BIGBALLON/Deep-learning-and-practices/tree/master/Lab3-NIN
- Xavier Initialization
  http://andyljones.tumblr.com/post/110998971763/an-explanation-of-xavier-initialization
- 《Self-Normalizing Neural Networks》
  https://arxiv.org/pdf/1706.02515.pdf