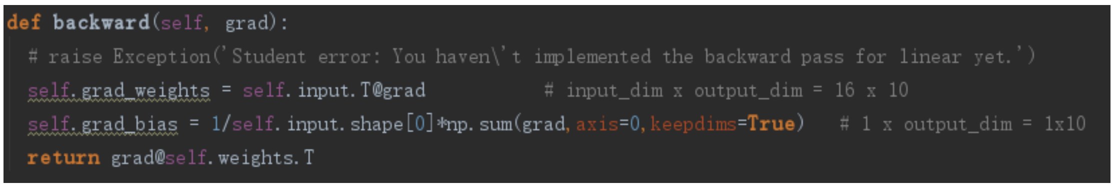
# Neural Network For Digit Identification

**Xindi Guo**

## 1. Implementing the Backward Pass for a Linear Layer.

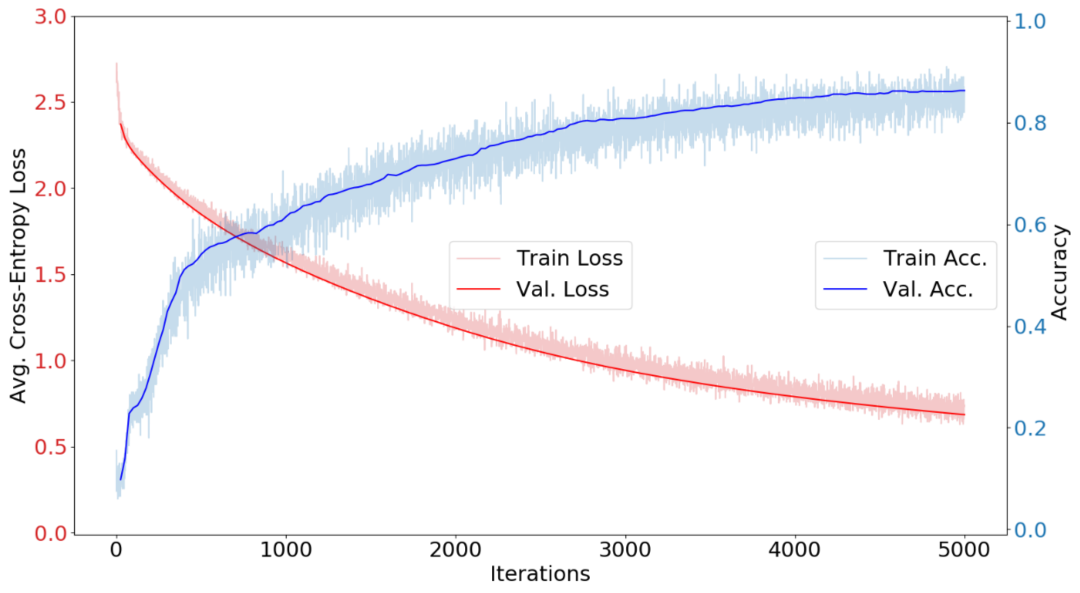
```
def backward(self,grad):
    self.grad_weights = self.input.T@grad
    self.grad_bias = 1/self.input.shape[0]*np.sum(grad,axis=0,keepdims=True)
    return grad@self.weights.T
```



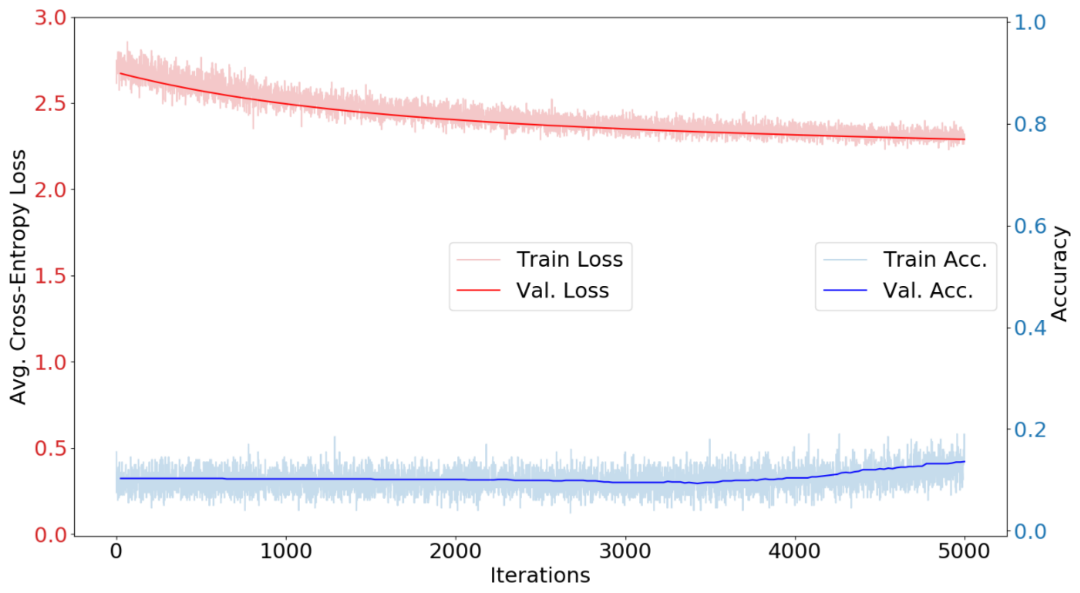
```
def backward(self, grad):
  # raise Exception('Student error: You haven\'t implemented the backward pass for linear yet.')
  self.grad_weights = self.input.T@grad            # input_dim x output_dim = 16 x 10
  self.grad_bias = 1/self.input.shape[0]*np.sum(grad,axis=0,keepdims=True)    # 1 x output_dim = 1x10
  return grad@self.weights.T
```

## 2. Learning Rate.

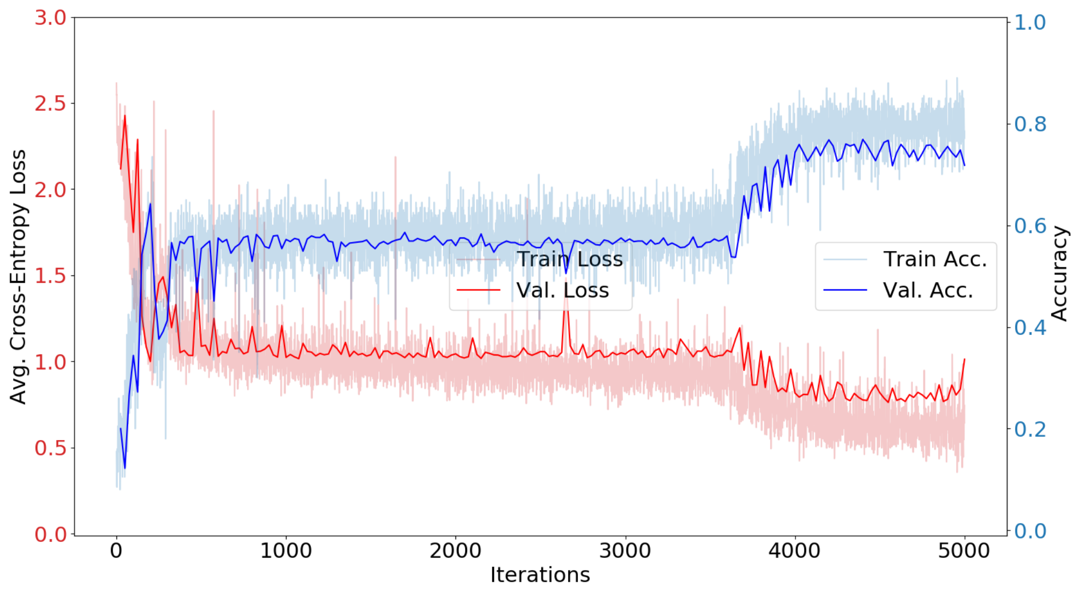
**Original plot：**



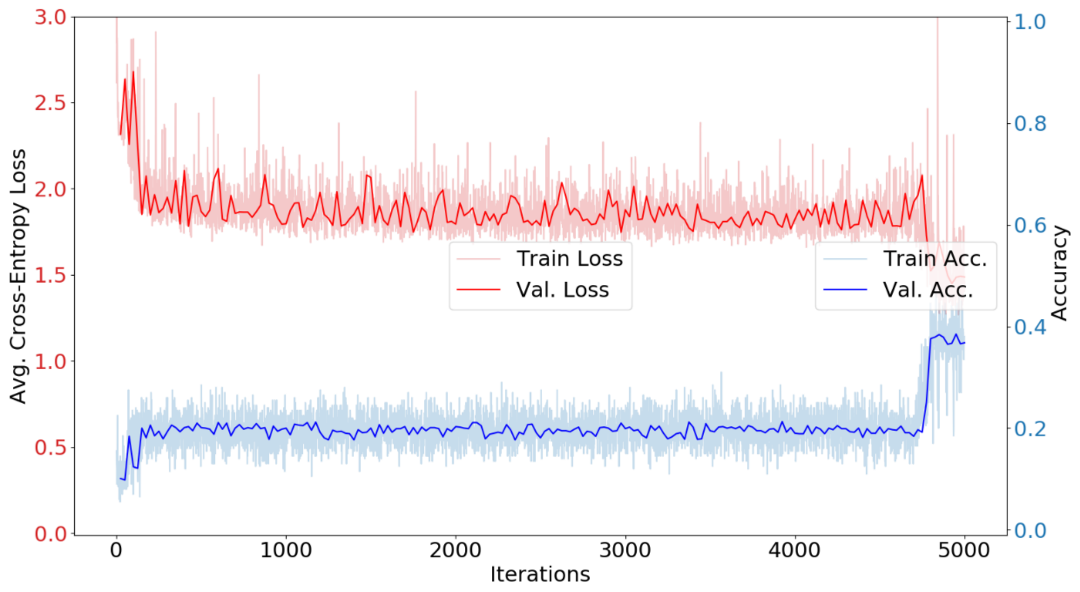
**Step size of 0.0001：**



**Step size of 5：**



**Step size of 10：**



**a) Compare and contrast the learning curves with your curve using the default parameters. What do you observe in terms of smoothness, shape, and what performance they reach?**

**step size is 0.0001**: the accuracy curve is smooth, the fluctuation is small and the validation accuracy is 13.6%.

**the step size is 5**: the accuracy of the model is improved rapidly, then keep stable for a long time, and finally improved to a better result and the validation accuracy is 71.8%.

**step size is 10**: the accuracy of the model is improved rapidly, then keep stable for a long time, and finally improved to a better result and the validation accuracy is 36.8%.

**b) For (a), what would you expect to happen if the max epochs were increased?**

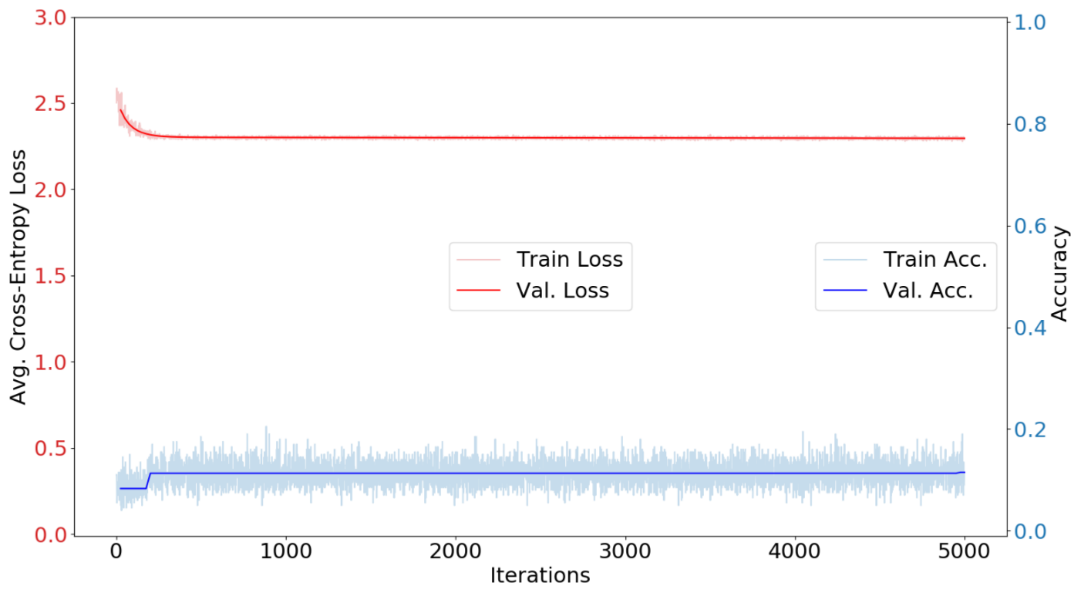
**step size is 0.0001**: the validation accuracy will be improved slowly.

**the step size is 5**: the validation accuracy may decline and fluctuate greatly**.**

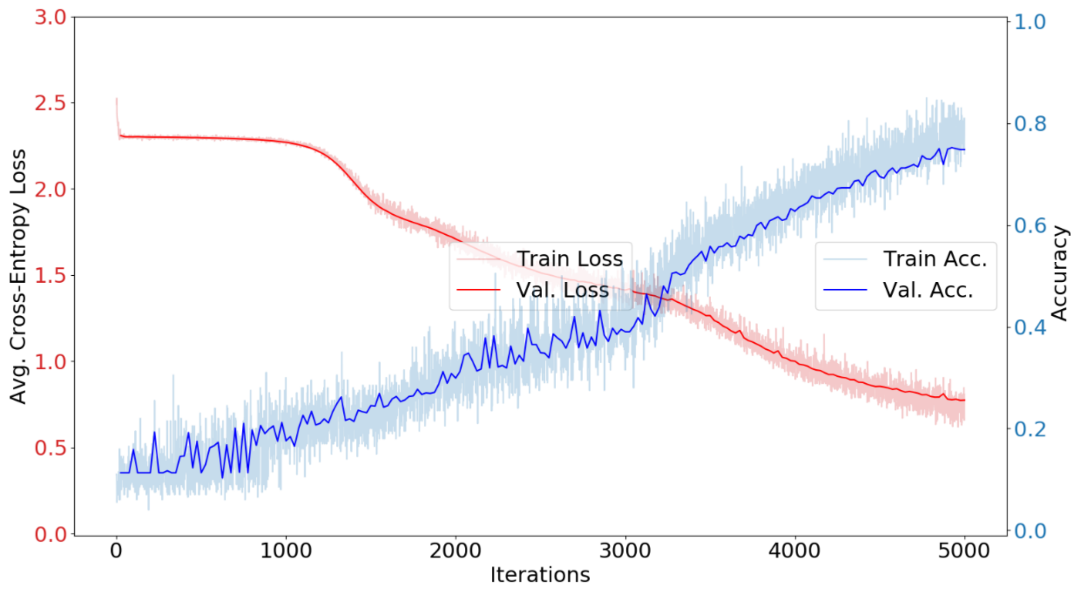
**step size is 10**: the validation accuracy may decline and fluctuate greatly.

## 3. ReLU's and Vanishing Gradients

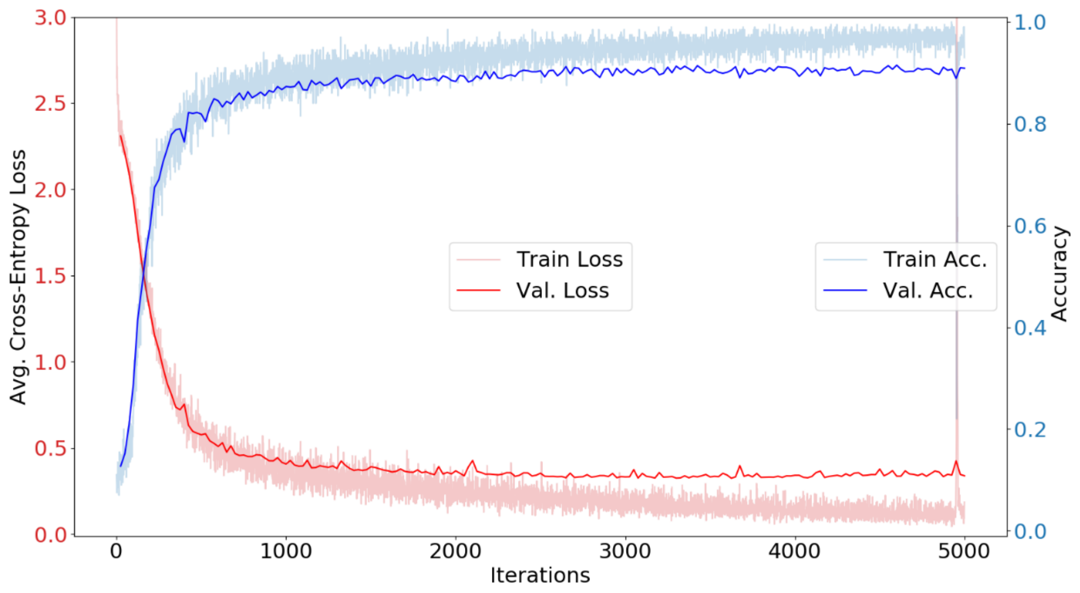
1. 5-layer with Sigmoid Activation (leave default values for other hyperparameters)



2. 5-layer with Sigmoid Activation with 0.1 step size (leave default values for other hyperparameters)



3. 5-layer with ReLU Activation (leave default values for other hyperparameters)



**a) Compare and contrast the learning curves you observe and the curve for the default parameters in terms of smoothness, shape, and what performance they reach. Do you notice any differences in the relationship between the train and validation curves in each plot?**

When using sigmoid function and small learning rate, the model accuracy curve rises slowly. When larger learning rate is adopted, the model accuracy improves faster. But ReLU function makes the model learns faster and converges faster.

The loss of the network is decreasing, the accuracy is increasing, and finally remains relatively stable. The validation loss is slightly higher than train loss, and the validation accuracy is slightly lower than train accuracy.

**b) If you observed increasing the learning rate in (2) improves over (1), why might that be?**

Larger learning rate makes the model learn better parameters, and too small learning rate keeps the model stay in the local optimal solution.

**c) If (3) outperformed (1), why might that be? Consider the derivative of the sigmoid and ReLU functions.**

For (1), When sigmoid function propagates backward, it is easy for gradient to disappear, because the derivatives at both ends tend to be zero and change too slowly.

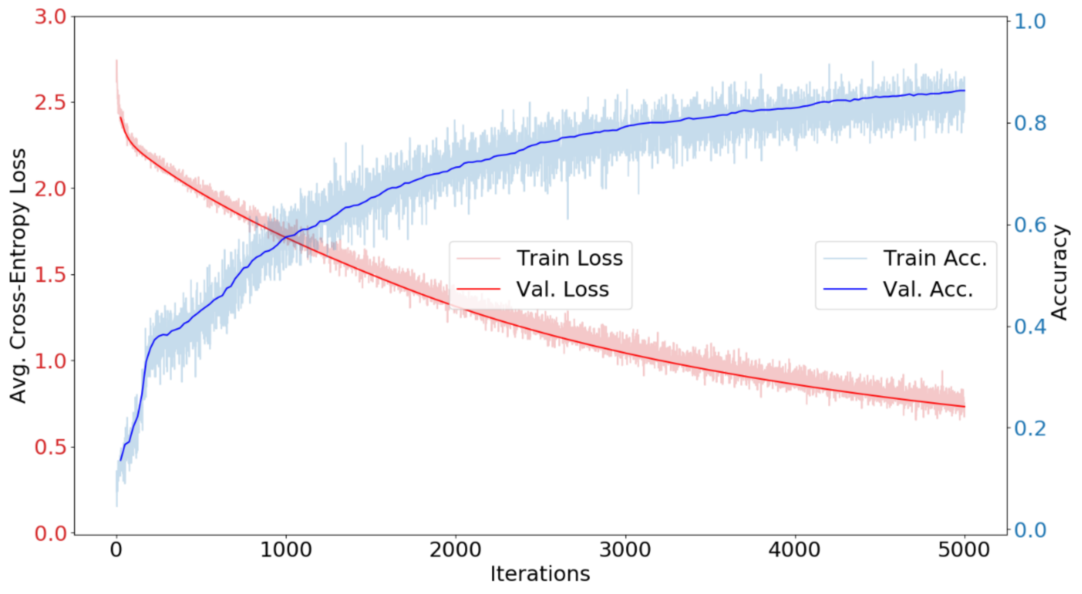
## 4. Measuring Randomness.

**Using the default hyperparameters, set the random seed to 5 different values and report the validation accuracies you observe after training. What impact does this randomness have on the certainty of your conclusions in the previous questions?**

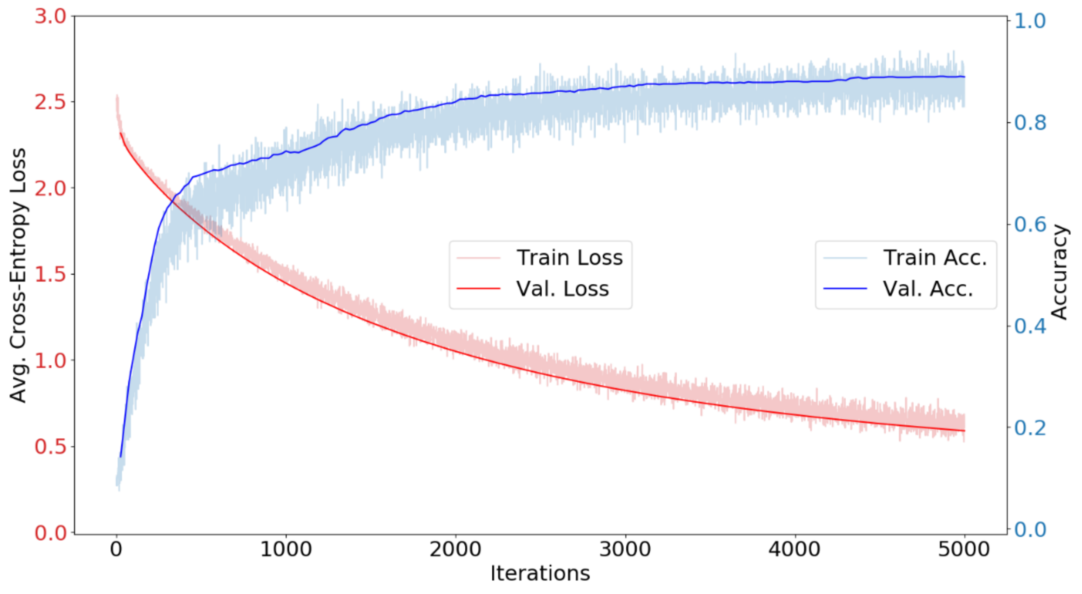
the validation accuracies get 86.3%, 89.0%, 87.8%, 89.5%, 88.2% when seed is 1, 50 ,200, 500, 1000.

The accuracy of the model will be affected by the initial values of the parameters, and the final results may have some deviation but maintain at a low level.

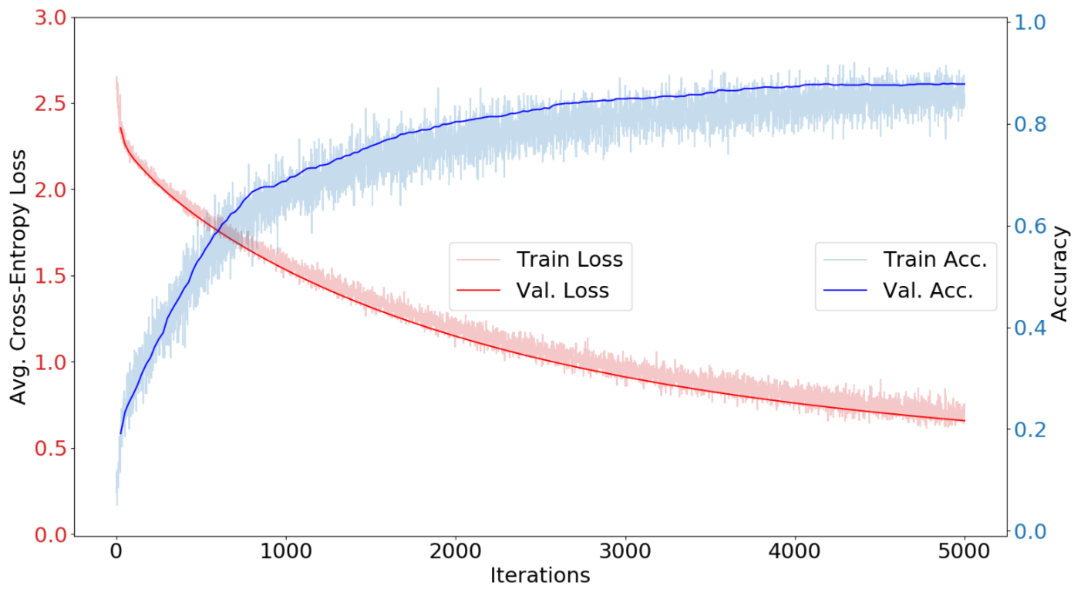
**Random seed is 1, the validation accuracy is 86.3%.**



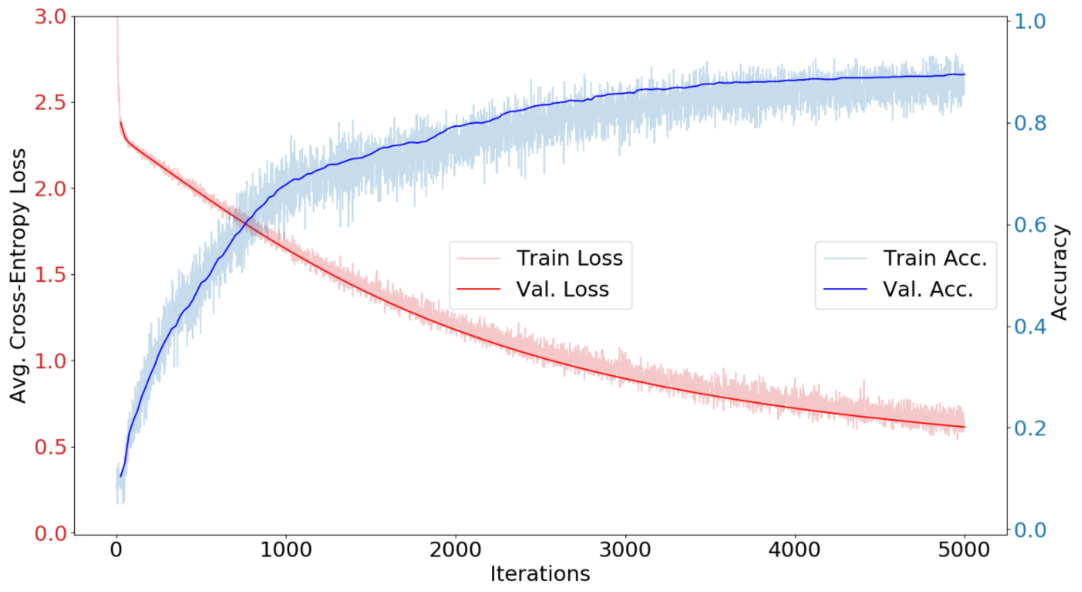
**Random seed is 50, the validation accuracy is 89.0%.**



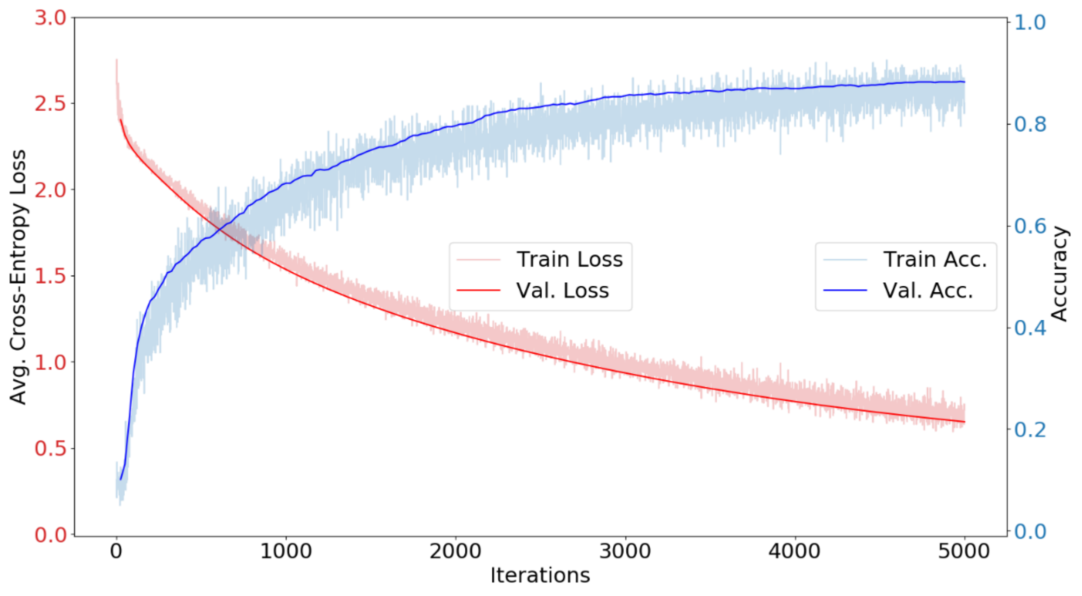
**Random seed is 200, the validation accuracy is 87.9%.**



**Random seed is 500, the validation accuracy is 89.5%.**



**Random seed is 1000, the validation accuracy is 88.2%.**



## 5. Kaggle Submission [8pt]

**Validation accuracy**: 93.5%
**Modifications** : the step size is 0.01,the batch size is 16 and the max_epochs is 400,and the random seed is 500.