Project Report 2
Empirical Research on CNN and CAE

Tingyuan Liang

April 7, 2018

Date Performed:	April 2, 2018
Instructor:	Professor Dit-Yan Yeung
TA:	Peter ZENG

1 Objectives

To acquire a better understanding of deep learning by using a public-domain software package called TensorFlow.

To evaluate the performance of convolutional neural network (CNN) and convolutional autoencoder (CAE) models by conducting empirical study on simple image data.

2 Major Tasks

The project consists of the following tasks:
1. To install and learn to use TensorFlow.
2. To train a CNN model for image classification using the data set provided.
3. To train a CAE model for feature learning using the data set provided.
4. To write up a project report.

3 Basic DEMO Video

Three YouTube videos are used to present the training procedure visually. For the sake of long training time, all of them are sped up for the convenience of reviewers. Note that different learning rates for CAE training are compared.
CNN DEMO:`https://www.youtube.com/watch?v=yarDP2YRYok`
CAE DEMO(Learning Rate = 0.1):`https://www.youtube.com/watch?v=I5Z8MNnHnKc`
CAE DEMO(Learning Rate = 0.001):`https://www.youtube.com/watch?v=6aa1Vx_Z8Hg`

5212-prj2-TensorFlow-CNN-DEMO
收看次數：0・4 小時前

5212-prj2-TensorFlow-CAE-DEMO
收看次數：0・4 小時前

5212-prj2-TensorFlow-CAE-DEMO
收看次數：3・12 小時前

Figure 1: DEMOs for Project2

4 Task 1: CNN model

In this project, the image classifier is based on stochastic gradient descent (SGD) learning with momentum optimization: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning, which has been involved in these project. SGD with momentum is method which helps accelerate gradients vectors in the right directions, thus leading to faster converging. [1]

4.1 The Experiment Settings of the CNN Model

According to 5-fold cross-validation, the major parameter settings of SGD Classifier with momentum are presented in Table 1, while other parameters stay the same as default.

Names	Parameter Settings
Classifier	SGD with Momentum
Activation Function	ReLU
Loss Function	Softmax Cross-Entropy Loss
η(Learning_Rate)	0.01
$beta$(Momentum Factor)	0.5
Batch Size	50
Epoch	4

Table 1: Experiment Settings of the CNN Model

Accuracies of models with different parameters in 5-fold cross-validation are presented in Table 2 and Figure 2. For those results, the size of mini-batch is 50 and the number of epochs is 4.

Learning Rate \ Beta	0.0	0.1	0.2	0.5	0.8	1.0
0.0001	0.0545	0.0884	0.0734	0.1476	0.5231	0.6492
0.0010	0.6728	0.7053	0.7165	0.7599	0.8021	0.2883
0.0100	0.8202	0.8194	0.8214	0.8222	0.8206	0.1925

Table 2: Accuracies of Models with Different Parameters in 5-Fold Cross-Validation

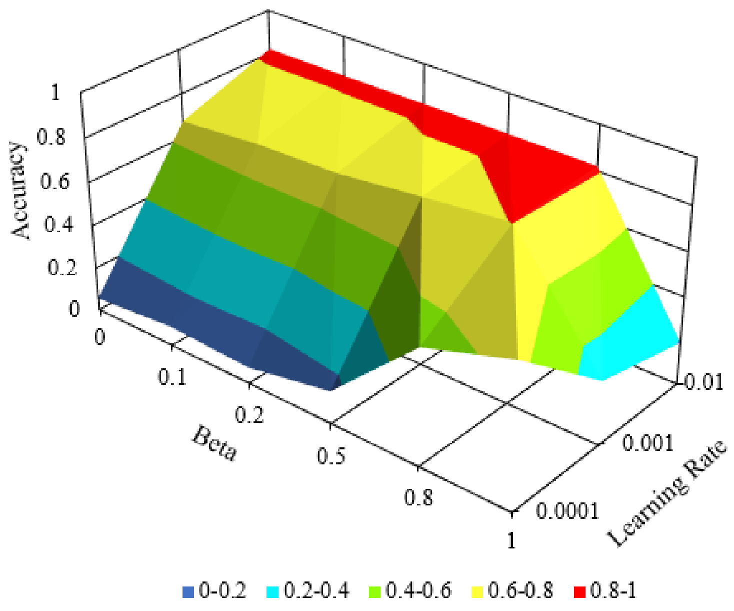

Figure 2: Accuracies of Models with Different Parameters in 5-Fold Cross-Validation

The hyperparameters of learning rate and β, the momentum factor, are tuned from the cross-validation to obtain high accuracy.

4.2 Explanation of Mini-Batch SGD with Momentum

SGD with momentum is method which helps accelerate gradients vectors in the right directions, thus leading to faster converging. [1] There is two definitions of SGD with momentum, which are pretty much just two different ways of writing the same equations.

First, is how Andrew Ng, defines it in his course on coursera: A momentum is defined as a moving average of gradients and then use it to update the weight of the network. This could written as follows:

$$M_t = \beta M_{t-1} + (1-\beta)\nabla_w L(W, X, y) \quad (1)$$
$$W = W - \eta M_t \quad (2)$$

Where M_t is the momentum, W is the weights in the network and $L(W, X, y)$ is the loss function of output. Note that β is the factor of momentum while η is the learning rate.

The other way, which is most popular way to write momentum update rules, is less intuitive and just omits $(1-\beta)$ term:

$$M_t = \beta M_{t-1} + \eta\nabla_w L(W, X, y) \quad (3)$$
$$W = W - M_t \quad (4)$$

This is pretty much identical to the first pair of equation, the only difference is that learning rate is scaled by $(1-\beta)$ factor.

In this project, the second description of SGD with momentum is implemented as a customized classifier and the related source code is shown below:

```
class ProjectOptimizer(optimizer.Optimizer):
    def __init__(self,
                 learning_rate=0.001,
                 beta=0.5,
                 use_locking=False,
                 name="ProjectOptimizer"):
        super(ProjectOptimizer, self).__init__(use_locking, name)
        self._lr = learning_rate
        self._beta = beta
        self._lr_t = None
        self._beta_t = None

    def _prepare(self):
        self._lr_t = ops.convert_to_tensor(self._lr, name="learning_rate")
        self._alpha_t = ops.convert_to_tensor(self._beta, name="alpha_t")
        self._beta_t = ops.convert_to_tensor(self._beta, name="beta_t")
    def _create_slots(self, var_list):
        for v in var_list:
            self._zeros_slot(v, "m", self._name)

    def _apply_dense(self, grad, var):
        lr_t = math_ops.cast(self._lr_t, var.dtype.base_dtype)
        beta_t = math_ops.cast(self._beta_t, var.dtype.base_dtype)

        eps = 1e-7  #cap for moving average

        m = self.get_slot(var, "m")
        m_t = m.assign(beta_t * m + lr_t * grad) # update the gradients with momentums
        var_update = state_ops.assign_sub(var, m_t)

        return control_flow_ops.group(*[var_update, m_t])
```

: Code Outline for SGD with Momentum

4.3 The Experiment Reseult of CNN Model

Based on TensorFlow, performance over time for different parameter settings, on both the test sets and training test, are obtained, as shown in Figure 3 and the accuracy and loss of CNN model are presented in Table 3. By changing some hyperparameters, other performance curves are obtained, as shown in Figure 4, Figure 5 and Figure 6.

	Accuracy	Loss	Training Time
Training Set	0.8612	0.4059	23min+30s
Evaluation Set	0.8225	0.5664	

Table 3: Accuracy and Loss of the CNN model(Standard)

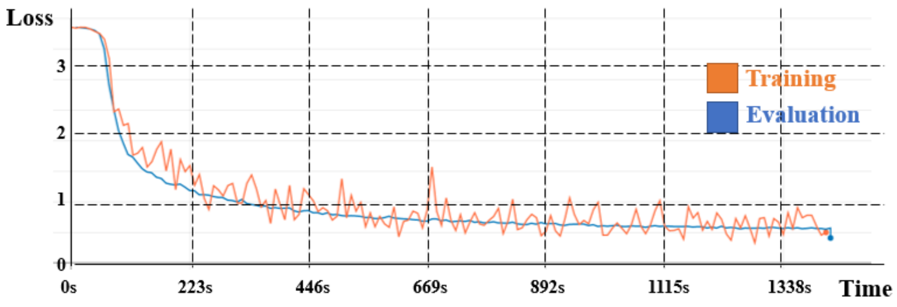

Figure 3: (Standard)$\eta = 0.01$, $\beta = 0.5$, $Batch\_size = 50$

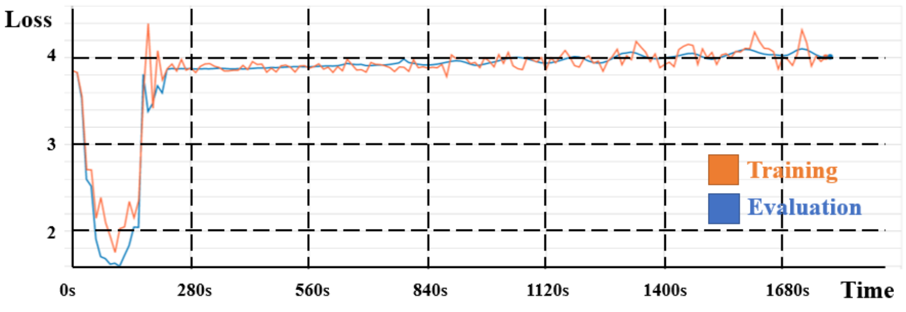

Figure 4: (Beta Changed)$\eta = 0.01$, $\beta = 1$, $Batch\_size = 50$

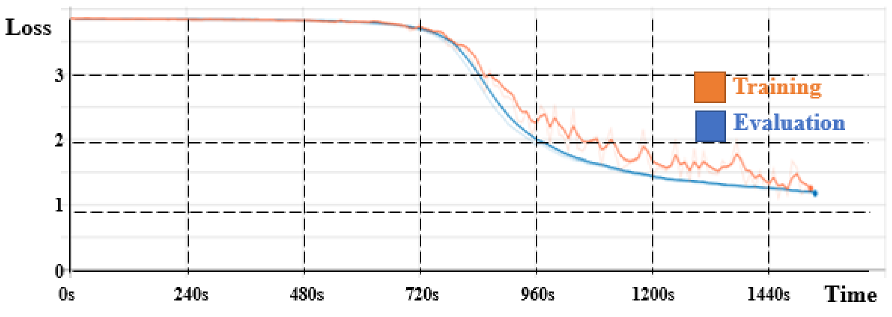

Figure 5: (η Changed)$\eta = 0.001$, $\beta = 0.5$, $Batch\_size = 50$

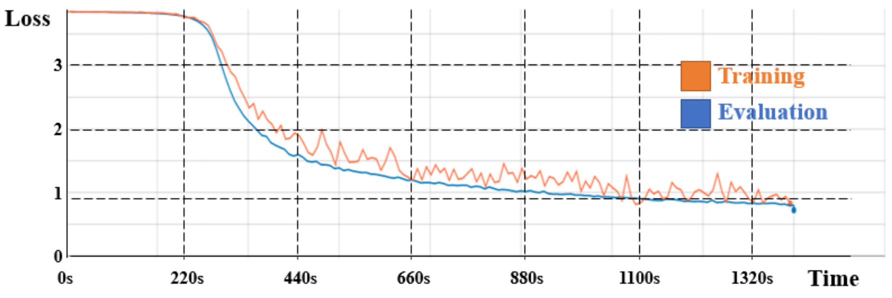

Figure 6: (Batch_size Changed)$\eta = 0.01$, $\beta = 0.5$, $Batch\_size = 200$

5 Task 2: CAE Model

Similar to Task 1, SGD learning with momentum optimization is applied to train a MAE.

5.1 Parameter Settings for CAE Training

According to 5-fold cross-validation, the major parameter settings of SGD Classifier with momentum are presented in Table 4, while other parameters stay the same as default.

Names	Parameter Settings
Classifier	SGD with Momentum
Activation Function	ReLU(for hidden layers) and Identity(for output layer)
Loss Function	MSE
Learning_Rate	0.1
beta(Momentum Factor)	0.5
Batch Size	20
Epoch	4

Table 4: Experiment Settings of the CNN Model

MSEs of models with different parameters in 5-fold cross-validation are presented in Table 5 and Figure 7. For those results, the size of mini-batch is 20 and the number of epochs is 4.

Beta / Learning Rate	0.0	0.1	0.2	0.5	1.0
0.001	0.0832	0.0742	0.0834	0.0664	0.0384
0.010	0.0256	0.0245	0.0256	0.0192	0.0573
0.100	0.0110	0.0109	0.0100	0.0087	0.0275

Table 5: MSEs of Models with Different Parameters in 5-Fold Cross-Validation

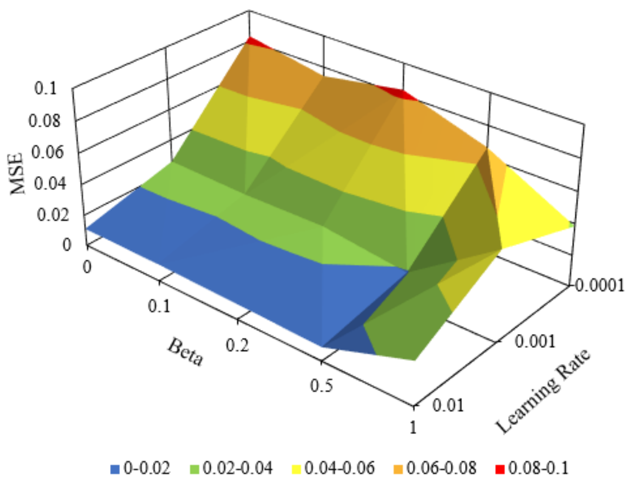

Figure 7: MSEs of Models with Different Parameters in 5-Fold Cross-Validation

5.2 The Experiment Reseult of CAE Model

Based on TensorFlow, the MSE loss of CAE model are presented in Table 6. Some reconstructed figures are shown in Figure 8. Moreover, their corresponding feature maps in the 7x7x2 encoded layers are demonstrated in Figure 9

	MSE Loss	Training Time
Training Set	0.0160	73min+02s
Evaluation Set	0.0152	

Table 6: MSEs of the CAE model

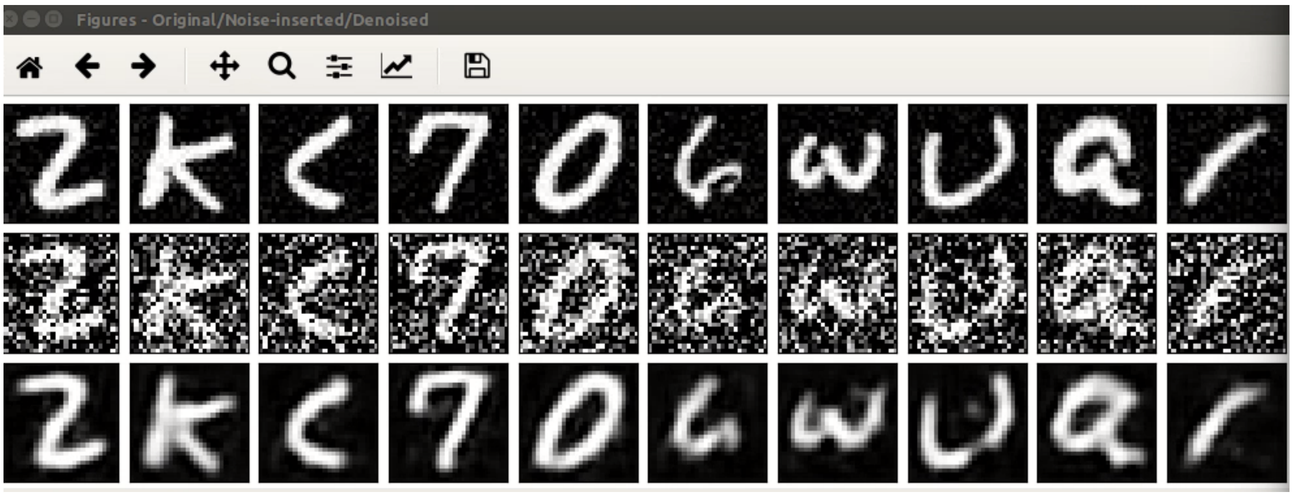

Figure 8: Top:Original – Middle:Noise Inserted – Bottom:Reconstructed(De-noised)

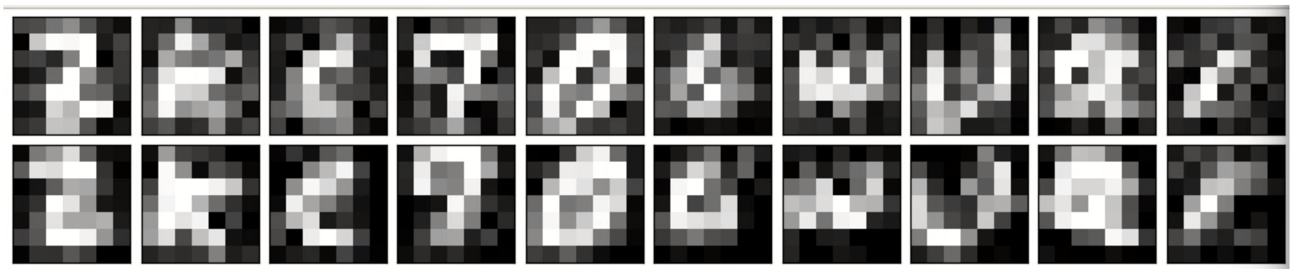

Figure 9: Feature Maps (Encoded Layers)

5.3 Reference Links

CAE:`https://github.com/mchablani/deep-learning/blob/master/autoencoder/Convolutional_Autoencoder.ipynb`

CNN:`https://www.tensorflow.org/get_started/custom_estimators`

custom-optimizer:`https://medium.com/bigdatarepublic/custom-optimizer-in-tensorflow-d5b41f75644a`

References

[1] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.