In [23]:

```
import torchvision
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
```

In [3]:

```
# Data download and preprocessing

DOWNLOAD_MNIST = True # If already download , set as False
train_data = torchvision.datasets.MNIST(
    root ='./mnist/',
    train = True , # this is training data
    # transform = torchvision.transforms.ToTensor(),
    download = DOWNLOAD_MNIST,
)
test_data = torchvision.datasets.MNIST(root ='./mnist/', train = False)

# change the features to numpy
X_train = train_data.train_data.numpy()
X_test = test_data.test_data.numpy()

# change the labels to numpy
Y_train = train_data.train_labels.numpy()
Y_test = test_data.test_labels.numpy()
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ./m
nist/MNIST/raw/train-images-idx3-ubyte.gz

Extracting ./mnist/MNIST/raw/train-images-idx3-ubyte.gz to ./mnist/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ./m
nist/MNIST/raw/train-labels-idx1-ubyte.gz

Extracting ./mnist/MNIST/raw/train-labels-idx1-ubyte.gz to ./mnist/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ./mn
ist/MNIST/raw/t10k-images-idx3-ubyte.gz

Extracting ./mnist/MNIST/raw/t10k-images-idx3-ubyte.gz to ./mnist/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ./mn
ist/MNIST/raw/t10k-labels-idx1-ubyte.gz

Extracting ./mnist/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./mnist/MNIST/raw


/usr/local/lib/python3.7/dist-packages/torchvision/datasets/mnist.py:75: UserW
arning: train_data has been renamed data
  warnings.warn("train_data has been renamed data")
/usr/local/lib/python3.7/dist-packages/torchvision/datasets/mnist.py:80: UserW
arning: test_data has been renamed data
  warnings.warn("test_data has been renamed data")
/usr/local/lib/python3.7/dist-packages/torchvision/datasets/mnist.py:65: UserW
arning: train_labels has been renamed targets
  warnings.warn("train_labels has been renamed targets")
/usr/local/lib/python3.7/dist-packages/torchvision/datasets/mnist.py:70: UserW
arning: test_labels has been renamed targets
  warnings.warn("test_labels has been renamed targets")
```

3(a)

In [24]:

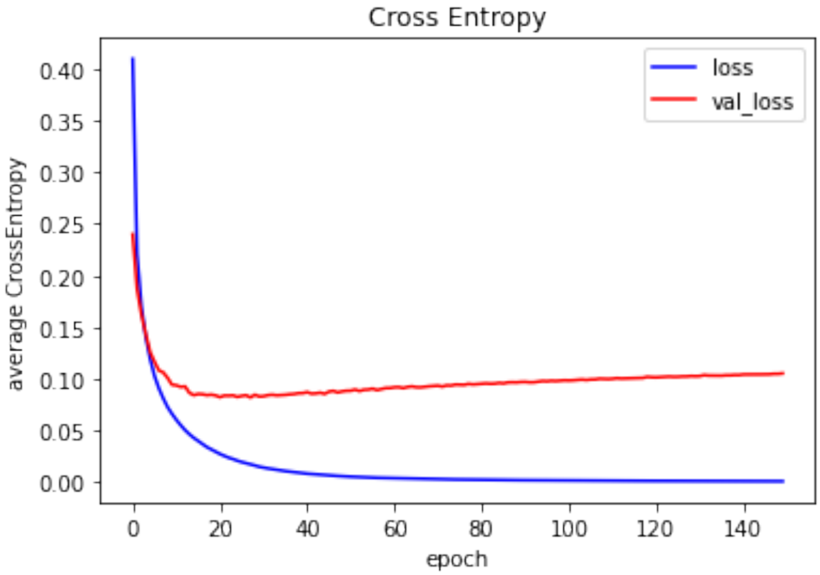
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import tensorflow.keras as keras
from keras.models import Sequential
from keras.layers import Dense
from keras.metrics import SparseCategoricalCrossentropy
from sklearn import preprocessing
```

In [5]:

```
mms = preprocessing.MinMaxScaler()
scaled_X_train = mms.fit_transform(X_train.reshape(60000,28*28))
X_train_new = scaled_X_train.reshape(60000, 784)
```

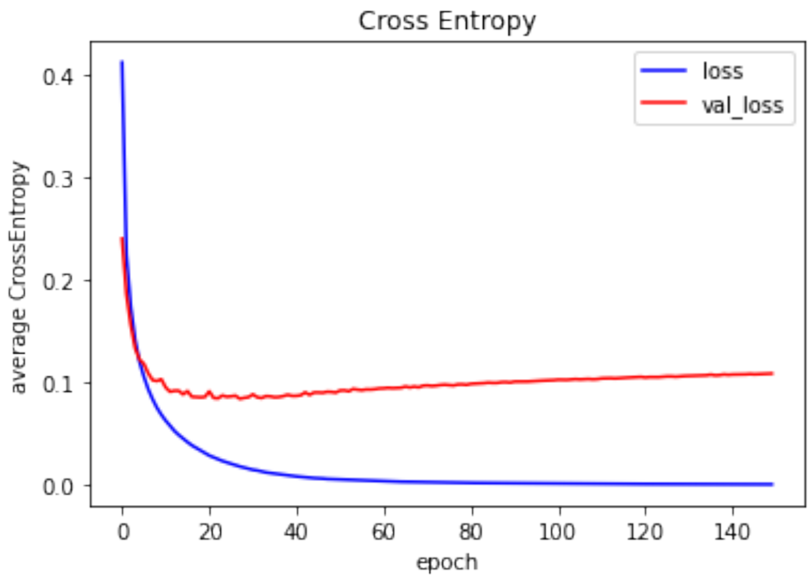
In [ ]:

```
np.random.seed(1)
sgd=tf.keras.optimizers.SGD(learning_rate=0.1)
model= Sequential()
model.add(Dense(100,activation='relu',input_shape=(784,)))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer=sgd,loss=tf.keras.losses.SparseCategoricalCrossentrop
history=model.fit(X_train_new,Y_train,epochs=150,validation_split=0.2,batch_s

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss")
plt.xlabel('epoch')
plt.ylabel('average CrossEntropy')
plt.title('Cross Entropy')
plt.legend(['loss','val_loss'],loc='best')
plt.show()
```



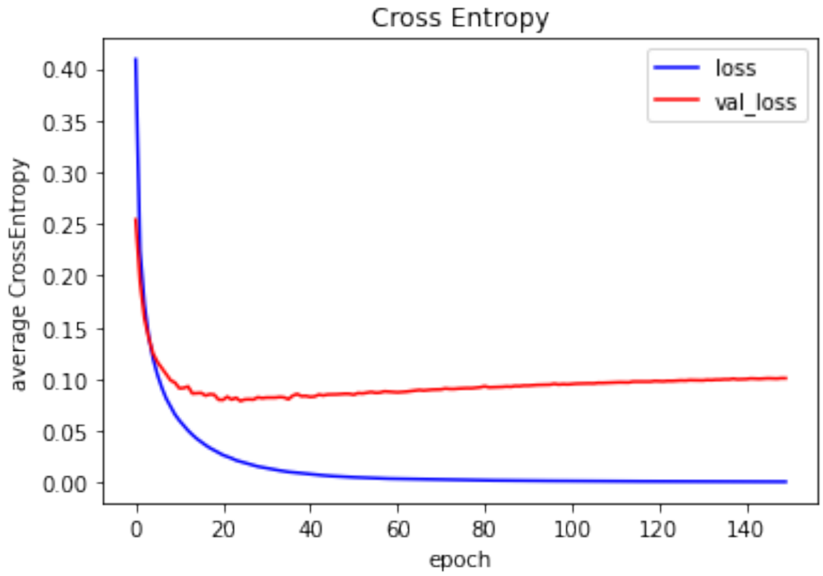
In [ ]:

```
np.random.seed(2)
sgd=tf.keras.optimizers.SGD(learning_rate=0.1)
model= Sequential()
model.add(Dense(100,activation='relu',input_shape=(784,)))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer=sgd,loss=tf.keras.losses.SparseCategoricalCrossentrop
history=model.fit(X_train_new,Y_train,epochs=150,validation_split=0.2,batch_s

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss")
plt.xlabel('epoch')
plt.ylabel('average CrossEntropy')
plt.title('Cross Entropy')
plt.legend(['loss','val_loss'],loc='best')
plt.show()
```



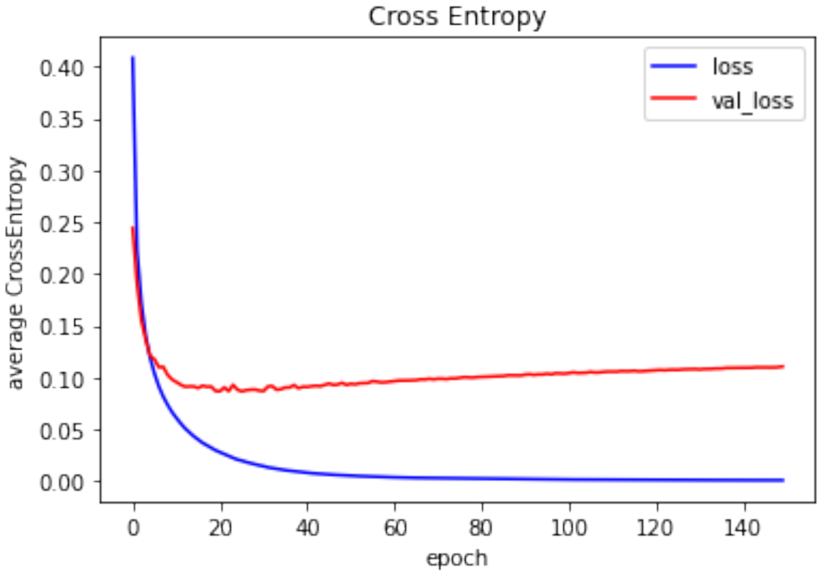
In [ ]:

```
np.random.seed(3)
sgd=tf.keras.optimizers.SGD(learning_rate=0.1)
model= Sequential()
model.add(Dense(100,activation='relu',input_shape=(784,)))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer=sgd,loss=tf.keras.losses.SparseCategoricalCrossentrop
history=model.fit(X_train_new,Y_train,epochs=150,validation_split=0.2,batch_s

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss")
plt.xlabel('epoch')
plt.ylabel('average CrossEntropy')
plt.title('Cross Entropy')
plt.legend(['loss','val_loss'],loc='best')
plt.show()
```



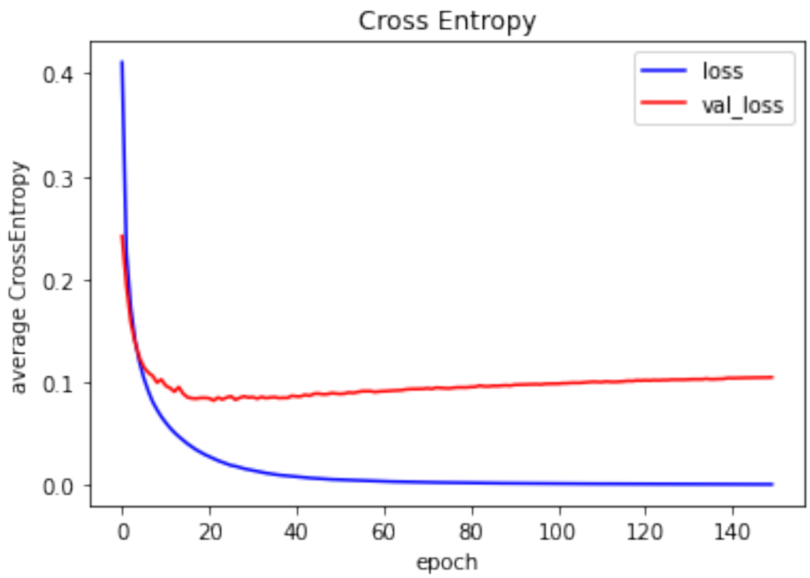
In [ ]:

```
np.random.seed(4)
sgd=tf.keras.optimizers.SGD(learning_rate=0.1)
model= Sequential()
model.add(Dense(100,activation='relu',input_shape=(784,)))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer=sgd,loss=tf.keras.losses.SparseCategoricalCrossentrop
history=model.fit(X_train_new,Y_train,epochs=150,validation_split=0.2,batch_s

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss")
plt.xlabel('epoch')
plt.ylabel('average CrossEntropy')
plt.title('Cross Entropy')
plt.legend(['loss','val_loss'],loc='best')
plt.show()
```



In [ ]:

```
np.random.seed(5)
sgd=tf.keras.optimizers.SGD(learning_rate=0.1)
model= Sequential()
model.add(Dense(100,activation='relu',input_shape=(784,)))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer=sgd,loss=tf.keras.losses.SparseCategoricalCrossentrop
history=model.fit(X_train_new,Y_train,epochs=150,validation_split=0.2,batch_s

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss")
plt.xlabel('epoch')
plt.ylabel('average CrossEntropy')
plt.title('Cross Entropy')
plt.legend(['loss','val_loss'],loc='best')
plt.show()
```

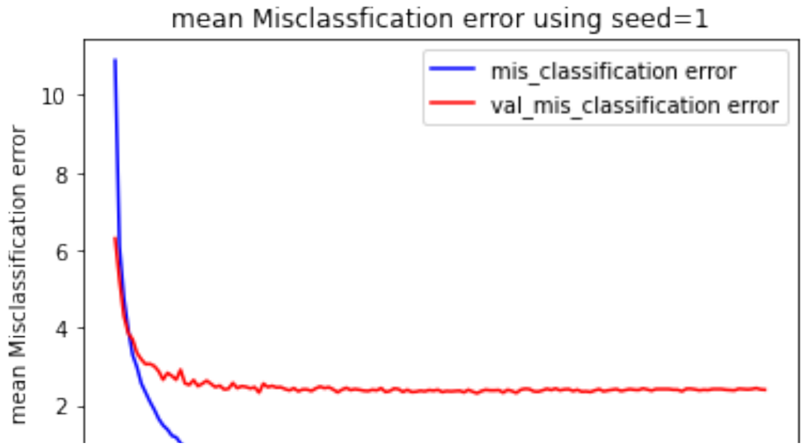


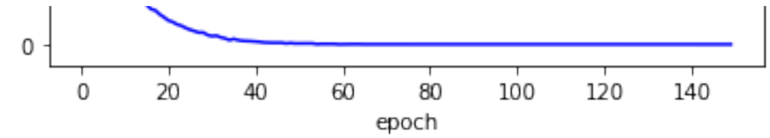
The validation error is higer than the training error. With the decrease of the training error, the validation error at first decrease and then a little increase, which suggests it's overfitting.

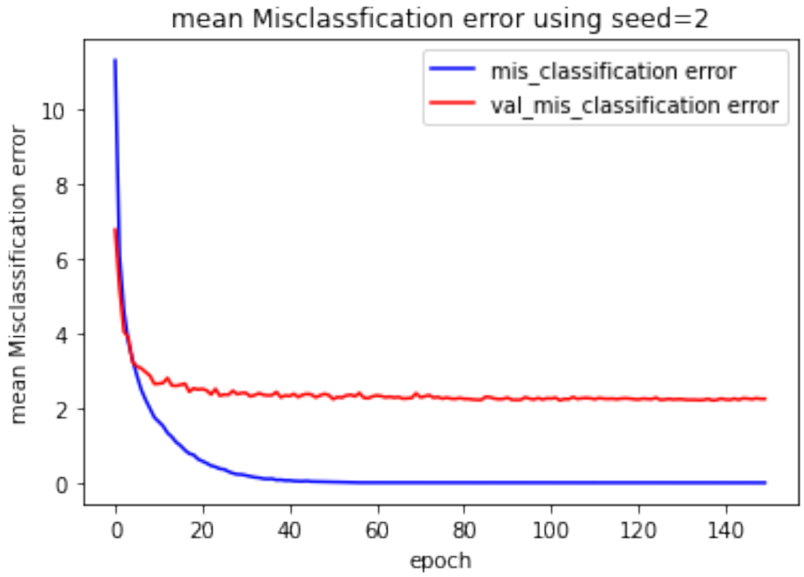
3(b)

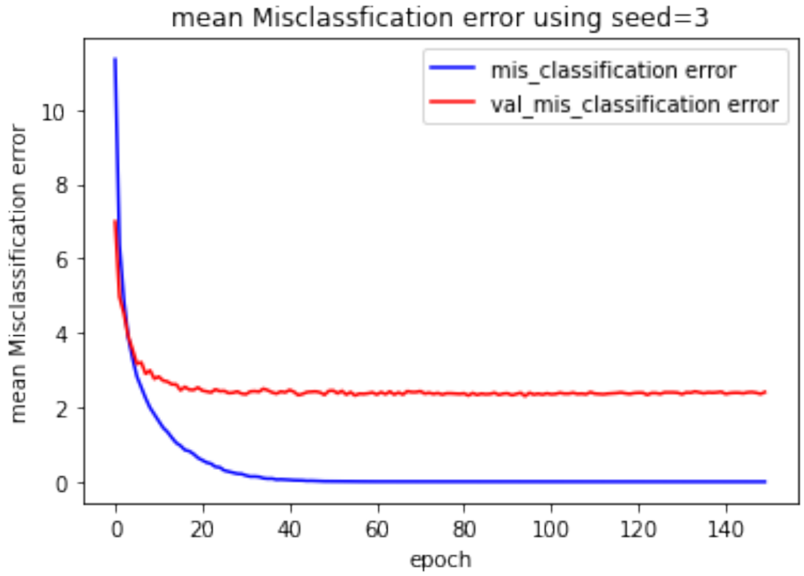
In [ ]:

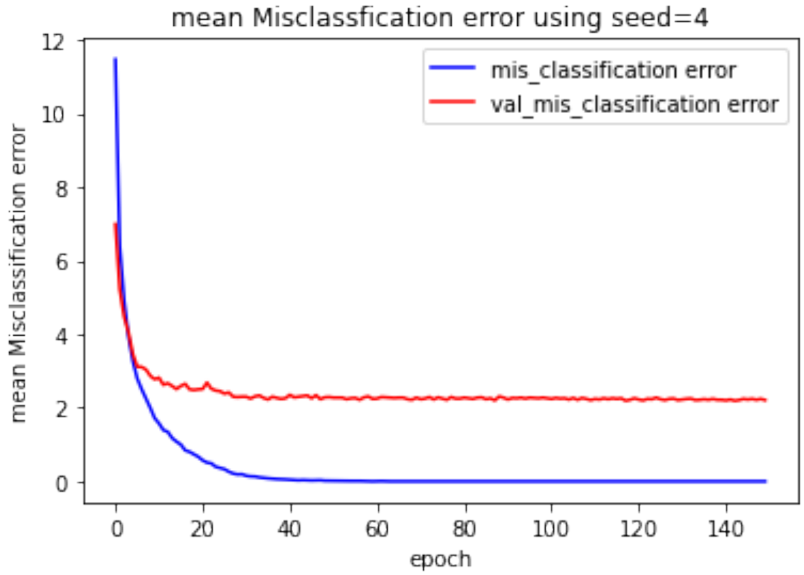
```
for seed in range(1,6):
  model = Sequential()
  model.add(Dense(100,activation='relu',input_shape=(784,)))
  model.add(Dense(10,activation='softmax'))
  np.random.seed(seed)
  model.compile(optimizer=sgd,loss=keras.losses.SparseCategoricalCrossentropy
  history=model.fit(X_train_new,Y_train,epochs=150,validation_split=0.2,batch_
  x = np.repeat(1,150)
  plt.plot((x-history.history['accuracy'])*100, color='b', label="mis_classif
  plt.plot((x-history.history['val_accuracy'])*100, color='r', label="val_mis_
  plt.xlabel('epoch')
  plt.ylabel('mean Misclassification error')
  plt.title('mean Misclassfication error using seed=%d'%(seed))
  plt.legend(['mis_classification error','val_mis_classification error'],loc=
  plt.show()
```

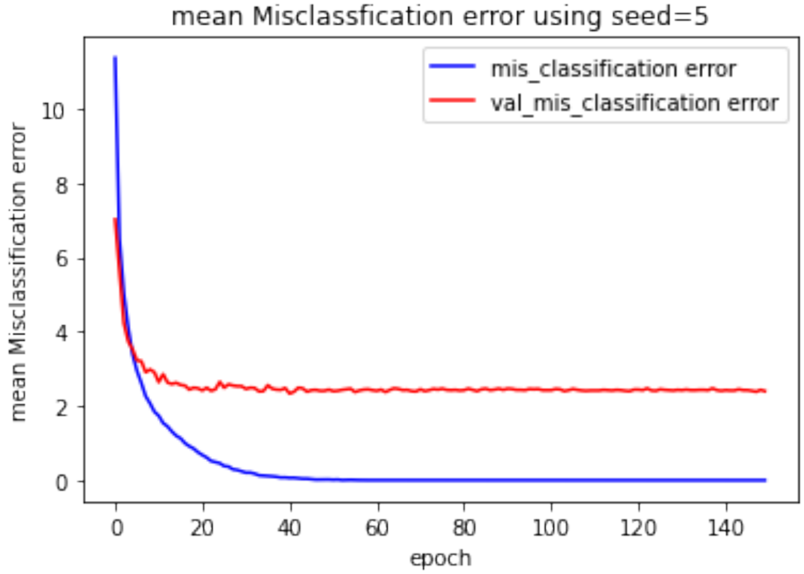



0
0
20
40
60
80
100
120
140
epoch


mean Misclassfication error using seed=2
mis_classification error
val_mis_classification error
mean Misclassification error
0
2
4
6
8
10
0
20
40
60
80
100
120
140
epoch


mean Misclassfication error using seed=3
mis_classification error
val_mis_classification error
mean Misclassification error
0
2
4
6
8
10
0
20
40
60
80
100
120
140
epoch



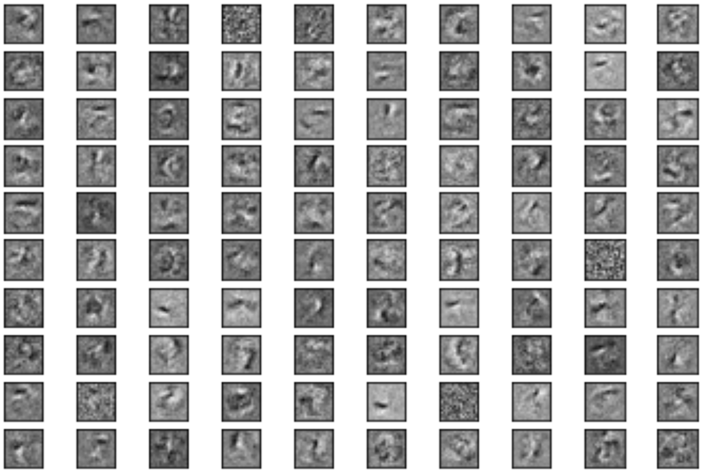


The validation error is still higher than the training error, but it is more certain compared to the behavior of the cross-entropy error function.

3(c)

In [ ]:

```
W = model.layers[0].get_weights()[0].reshape(28,28,100)
for i in range(W.shape[2]):
  plt.subplot(10, 10, i + 1)
  plt.imshow(W[:, :, i], cmap = 'gray')
  plt.xticks([])
  plt.yticks([])
plt.show()
```



The plot shows that it is uncorrelated and not too noisy, as well as given some curvy shapes.

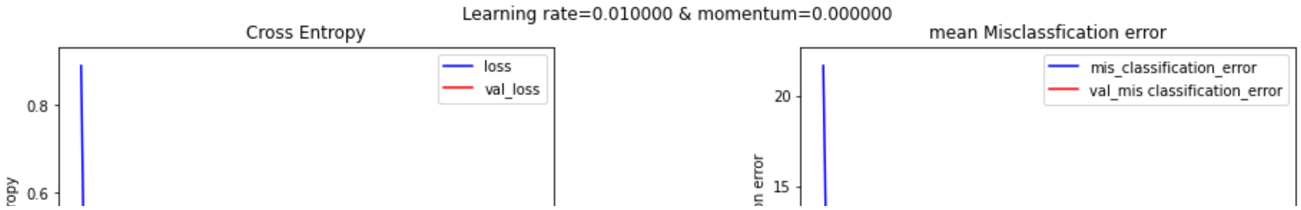
3(d)

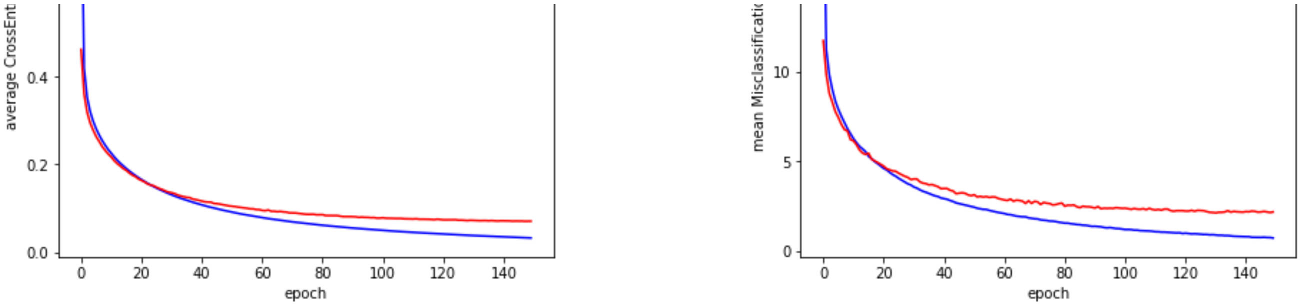
In [6]:

```
scaled_X_test = mms.fit_transform(X_test.reshape(10000,28*28))
X_test_new = scaled_X_test.reshape(10000, 784)
```

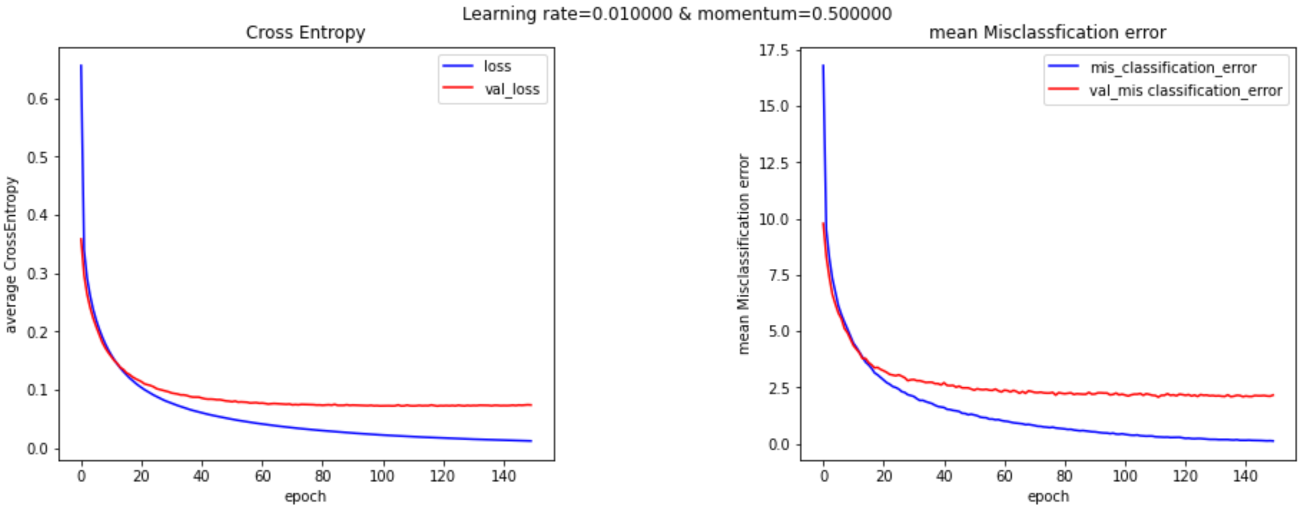
In [7]:

```
seed=3

for lr in [0.01, 0.1, 0.2, 0.5]:
  for momentum in [0, 0.5, 0.9]:
    sgd=keras.optimizers.SGD(learning_rate = lr, momentum = momentum)
    np.random.seed(seed)

    #Create single layer ANN
    model = Sequential()
    #Adding Hidden layer
    model.add(Dense(units=100, activation='relu', input_shape=(784,)))
    #Adding Output layer
    model.add(Dense(units=10, activation='softmax'))
    #Compiling the model
    model.compile(optimizer=sgd, loss=keras.losses.SparseCategoricalCrossentr
                  metrics=['accuracy'])
    history = model.fit(X_train_new, Y_train, epochs=150,
                        validation_data=(X_test_new,Y_test), batch_size=64, v

    #Plot
    f, ax = plt.subplots(1, 2, figsize=(15, 5))
    plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.5,
                    hspace=0.5)
    ax[0].plot(history.history['loss'], color='blue', label="Training loss")
    ax[0].plot(history.history['val_loss'], color='red', label="validation lo
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('average CrossEntropy')
    ax[0].set_title('Cross Entropy')
    ax[0].legend(['loss','val_loss'], fontsize = 10)

    x = np.repeat(1, 150)
    ax[1].plot((x - history.history['accuracy'])*100, color = 'blue', label =
    ax[1].plot((x - history.history['val_accuracy'])*100, color = 'red', labe
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('mean Misclassification error')
    ax[1].set_title('mean Misclassfication error')
    ax[1].legend(['mis_classification_error', 'val_mis classification_error']

    f.suptitle('Learning rate=%2f & momentum=%2f'%(lr, momentum))
    plt.show()
    print("The misclassification error for test set is %3f"%(((x - history.hi
```

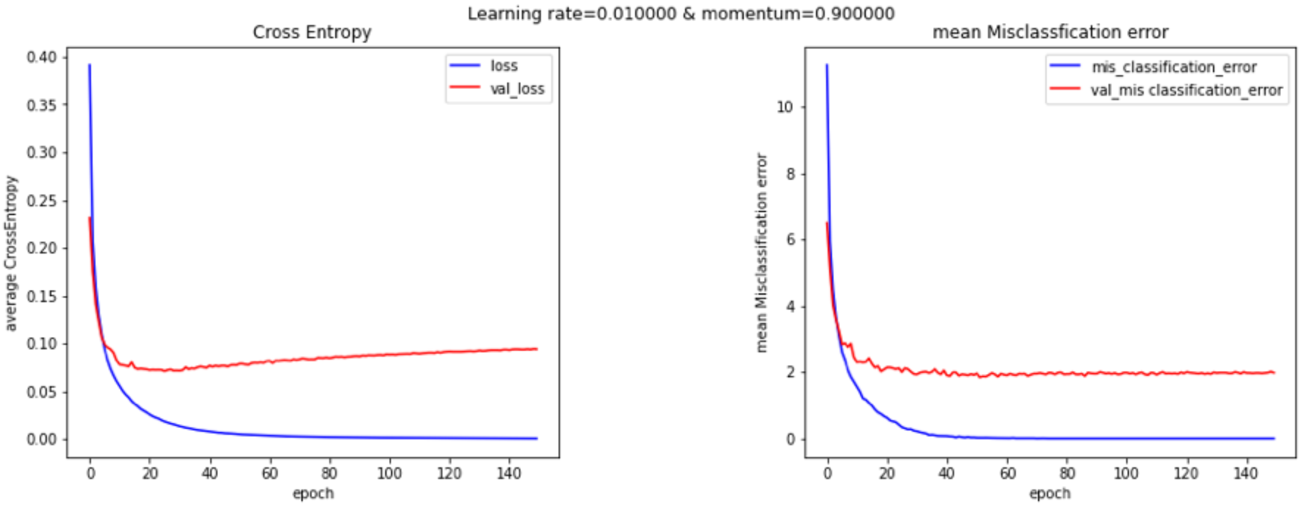




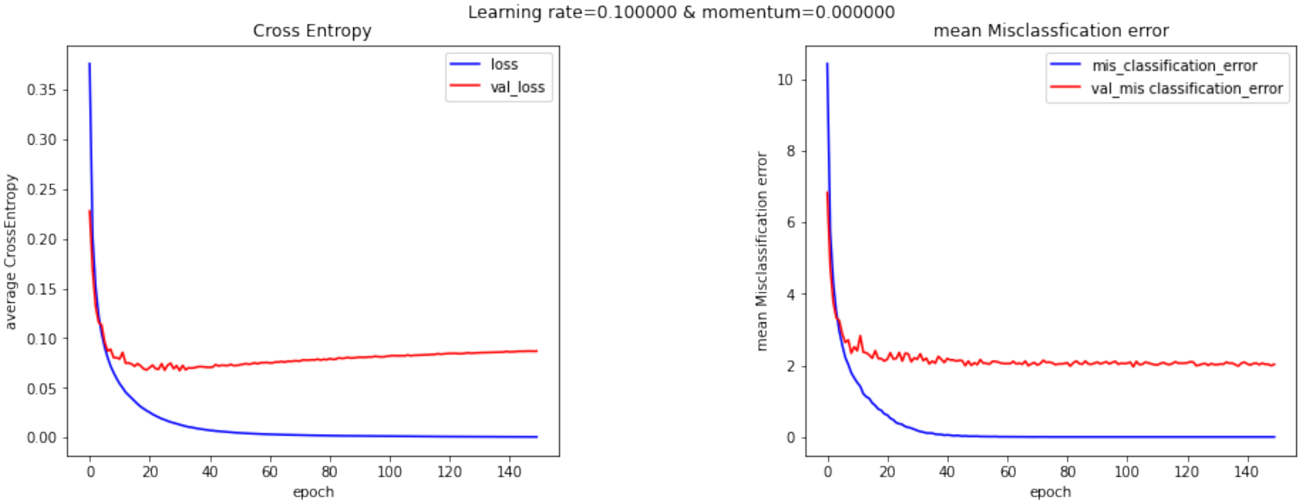
```
The misclassification error for test set is 6.230003%.
```



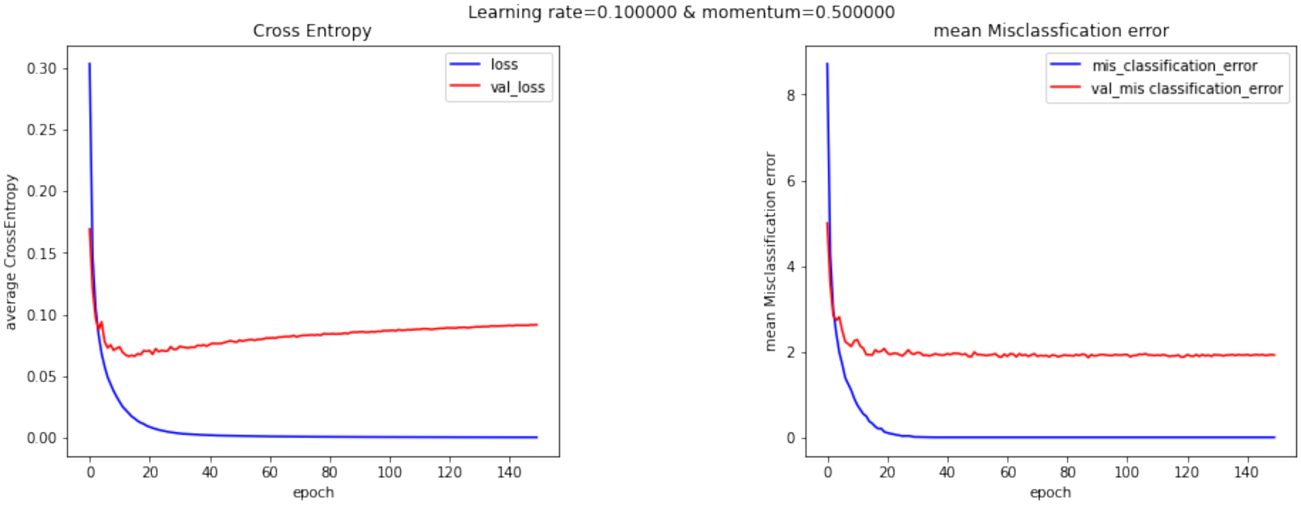
```
The misclassification error for test set is 4.640001%.
```



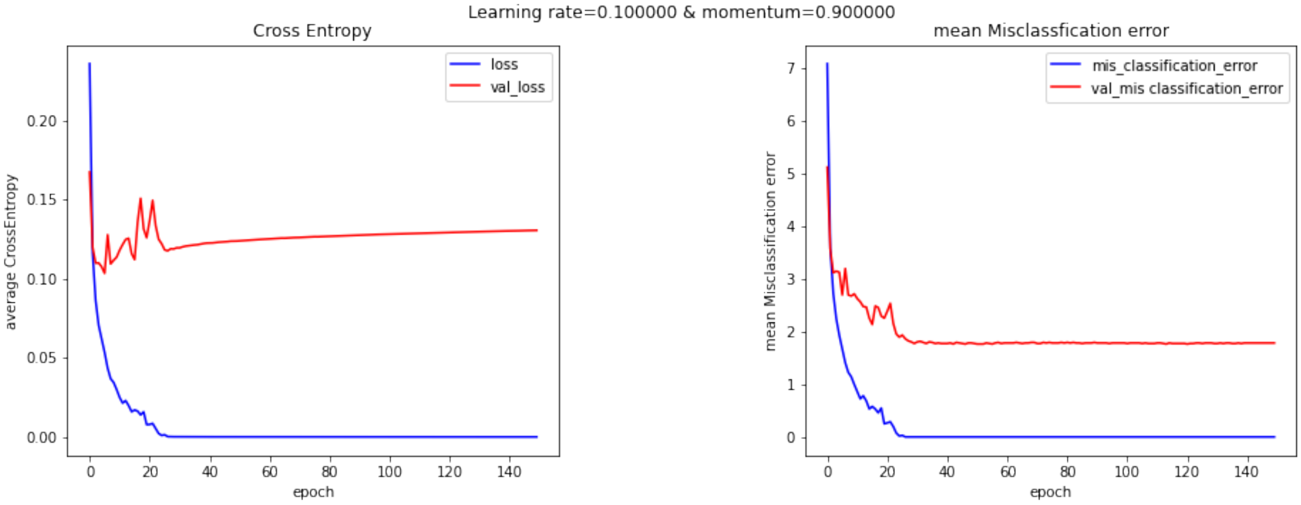
```
The misclassification error for test set is 2.450001%.
```



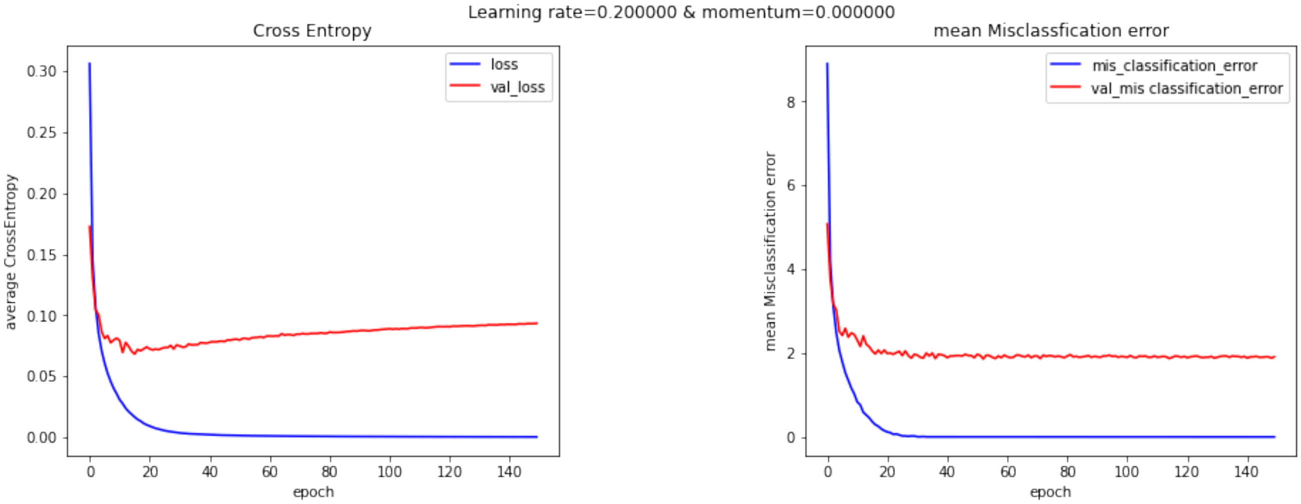
```
The misclassification error for test set is 2.520001%.
```



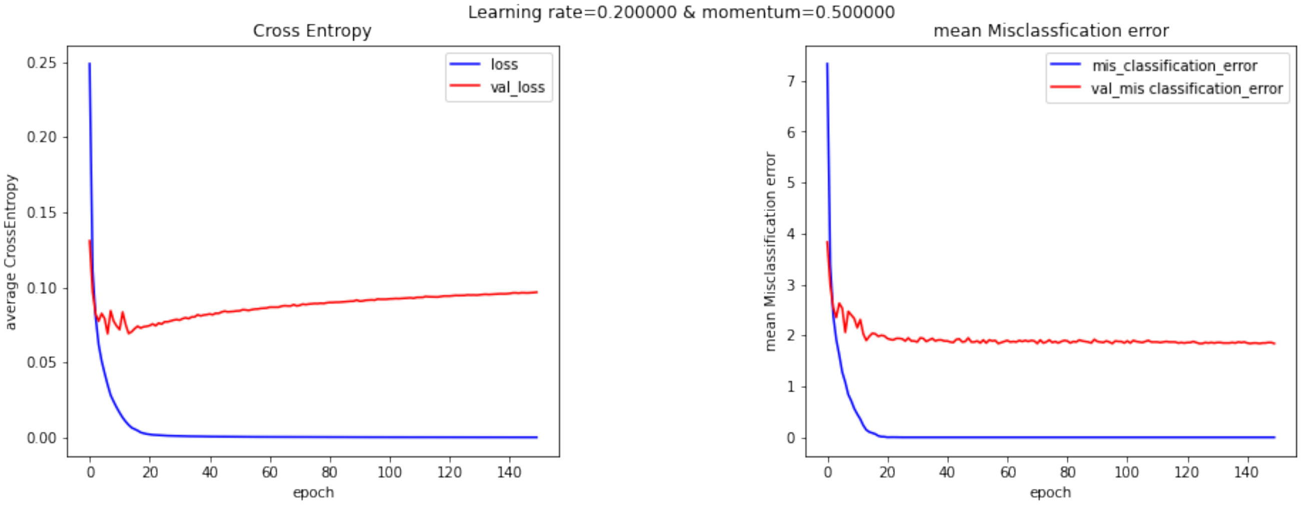
```
The misclassification error for test set is 2.249998%.
```



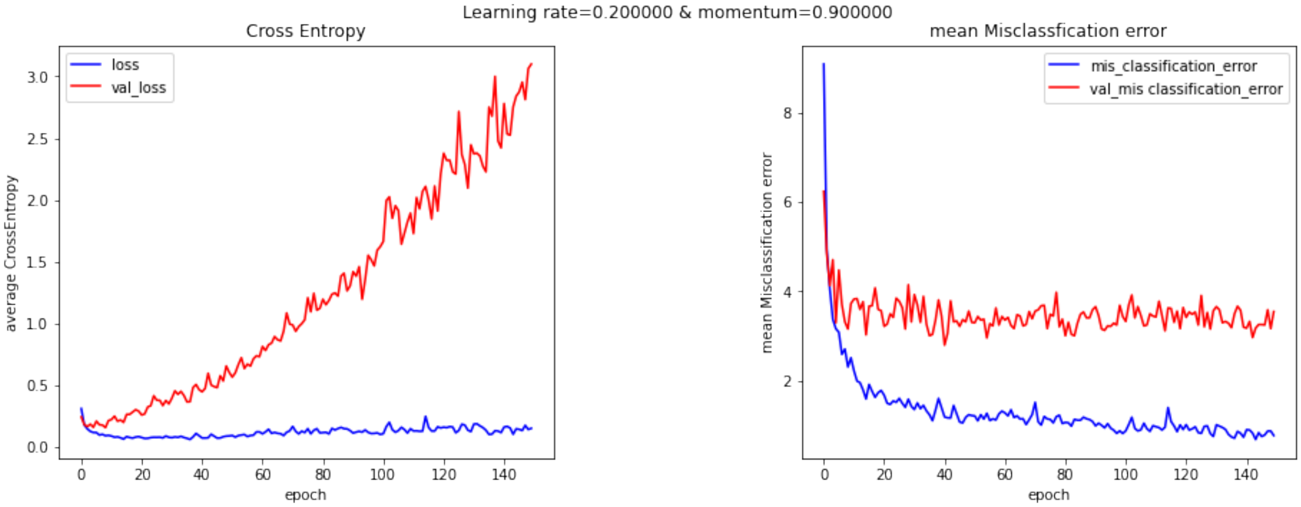
```
The misclassification error for test set is 2.710003%.
```



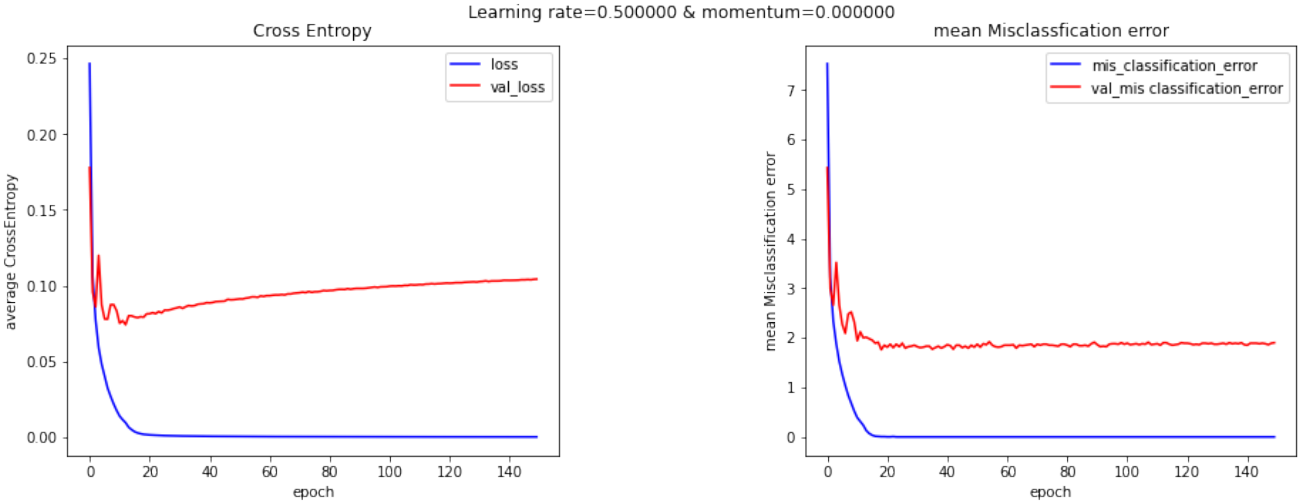
```
The misclassification error for test set is 2.440000%.
```



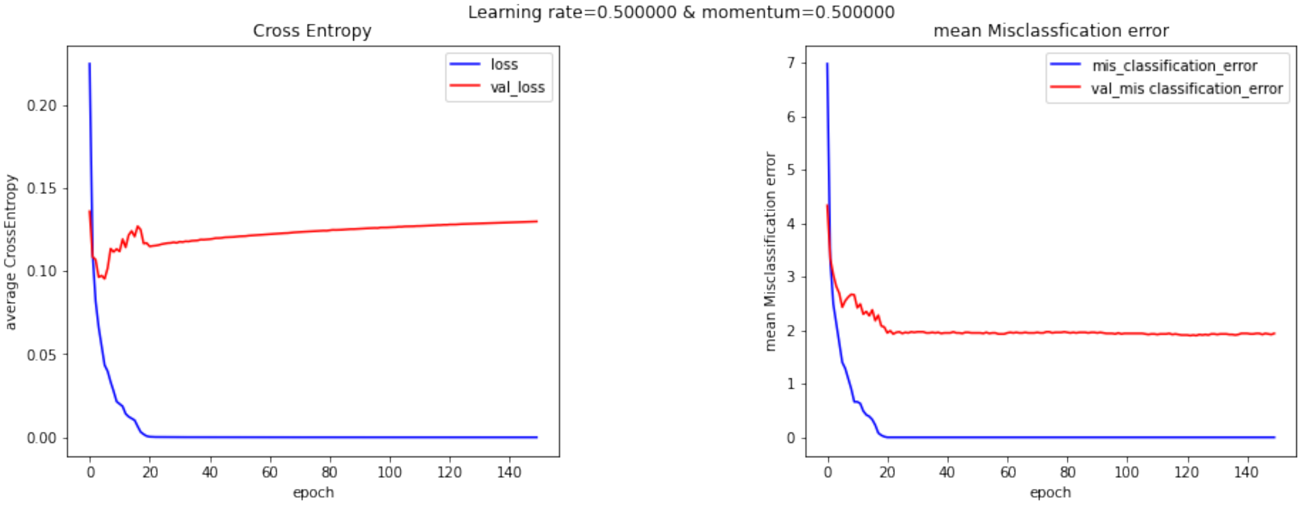
```
The misclassification error for test set is 2.329999%.
```



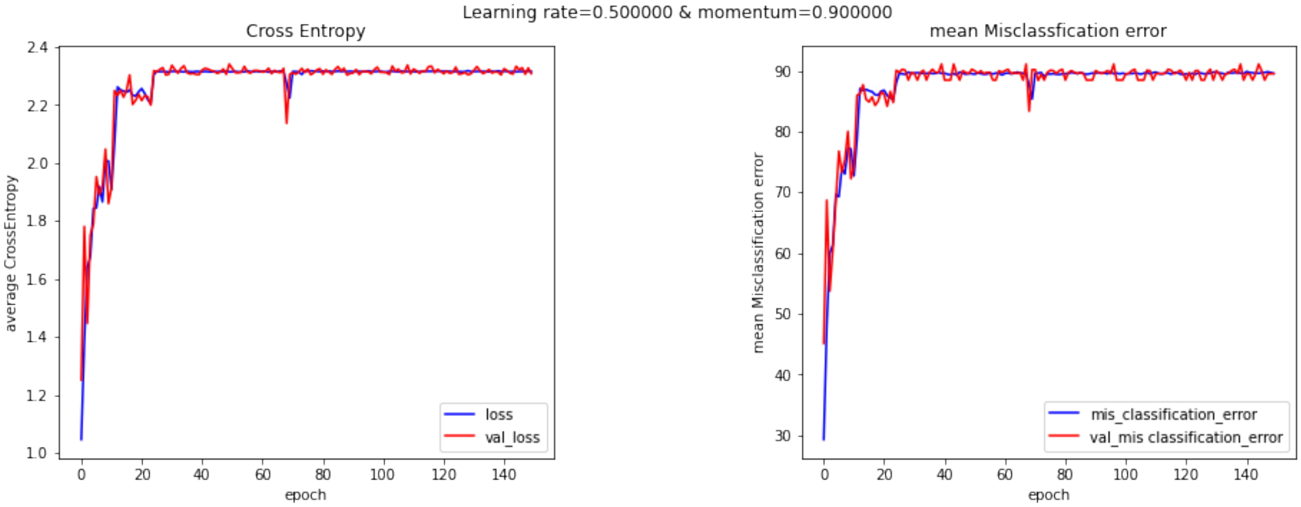
```
The misclassification error for test set is 3.729999%.
```



```
The misclassification error for test set is 2.319998%.
```



```
The misclassification error for test set is 2.660000%.
```



```
The misclassification error for test set is 72.229999%.
```

From plots,if fix the learning rate,we can see that when momentum is increasing, the error rate in most of cases is likely decreasing. If fix the momentum, we can see that when learing rate is increasing, the error rate becomes more variation. The best value of these parameters seems like learning rate = 0.1 and momentum = 0.5,

4(a)

In [8]:

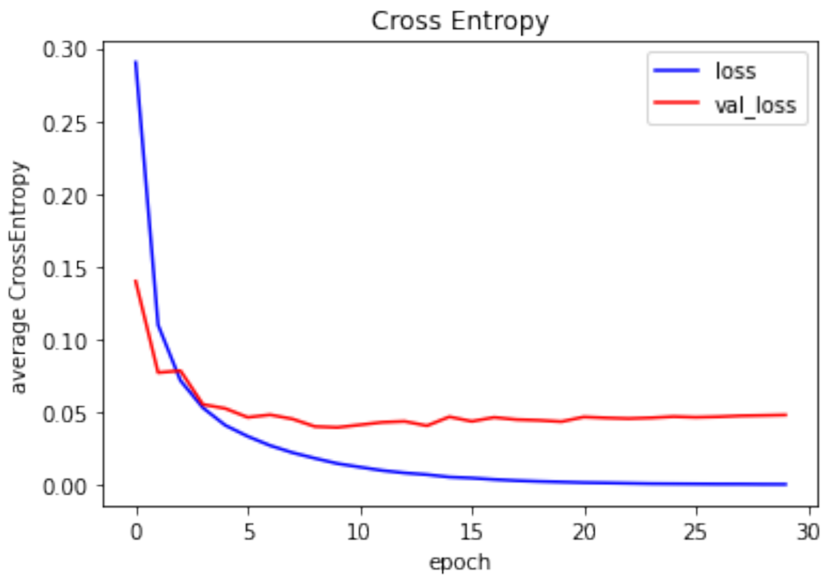
```
#Since 2-D space, reshape the data
X_train_new1 = X_train_new.reshape(X_train.shape[0], 28, 28, 1)
X_test_new1 = X_test_new.reshape(X_test.shape[0], 28, 28, 1)
```

In [9]:

```
from keras.layers import Conv2D,MaxPooling2D,Flatten
```

In [ ]:

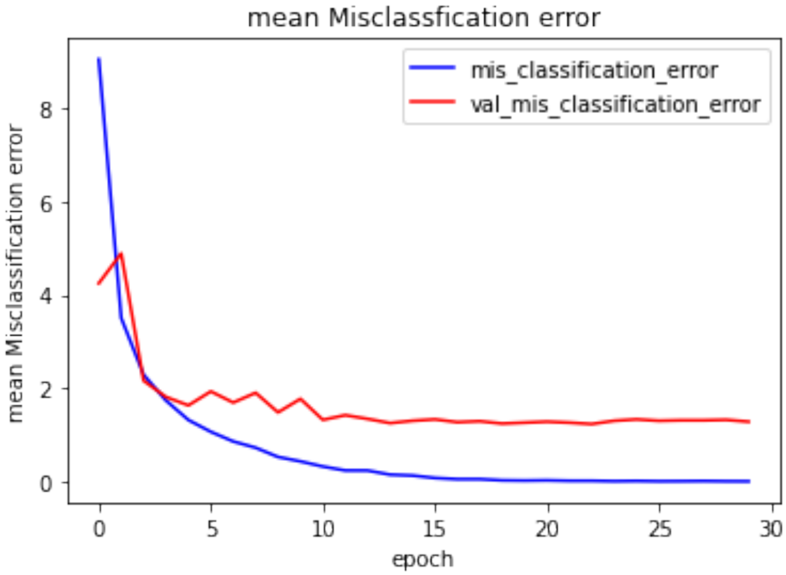
```
np.random.seed(3)
sgd = keras.optimizers.SGD(learning_rate = 0.1)
#Create single layer NN
model = Sequential()
#Adding Hidden layer
model.add(Conv2D(32, (3, 3), padding = 'same', activation='relu', input_shape
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(units = 100, activation = 'relu'))
model.add(Dense(units = 10, activation = 'softmax'))
#Compiling the model
model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCrossentr
history = model.fit(X_train_new1, Y_train, epochs = 30, validation_data = (X_

#Plot
plt.plot(history.history['loss'], color = 'blue', label = "Training loss")
plt.plot(history.history['val_loss'], color = 'red', label = "validation loss
plt.xlabel('epoch')
plt.ylabel('average CrossEntropy')
plt.title('Cross Entropy')
plt.legend(['loss','val_loss'], fontsize = 10)

plt.show()
```



4(b)

In [ ]:

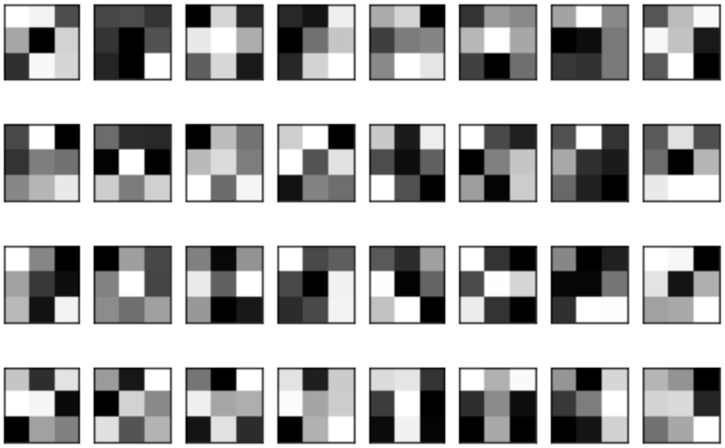
```
np.random.seed(3)
sgd = keras.optimizers.SGD(learning_rate = 0.1)
#Create single layer NN
model = Sequential()
#Adding Hidden layer
model.add(Conv2D(32, (3, 3), padding = 'same', activation='relu', input_shape
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(units = 100, activation = 'relu'))
model.add(Dense(units = 10, activation = 'softmax'))
#Compiling the model
model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCrossentr
history = model.fit(X_train_new1, Y_train, epochs = 30, validation_data = (X_

#Plot
x = np.repeat(1, 30)
plt.plot((x - history.history['accuracy'])*100, color = 'blue', label = "mis_
plt.plot((x - history.history['val_accuracy'])*100, color = 'red', label = "v
plt.xlabel('epoch')
plt.ylabel('mean Misclassification error')
plt.title('mean Misclassfication error')
plt.legend(['mis_classification_error', 'val_mis_classification_error'], font
plt.show()
```



4(c)

In [13]:

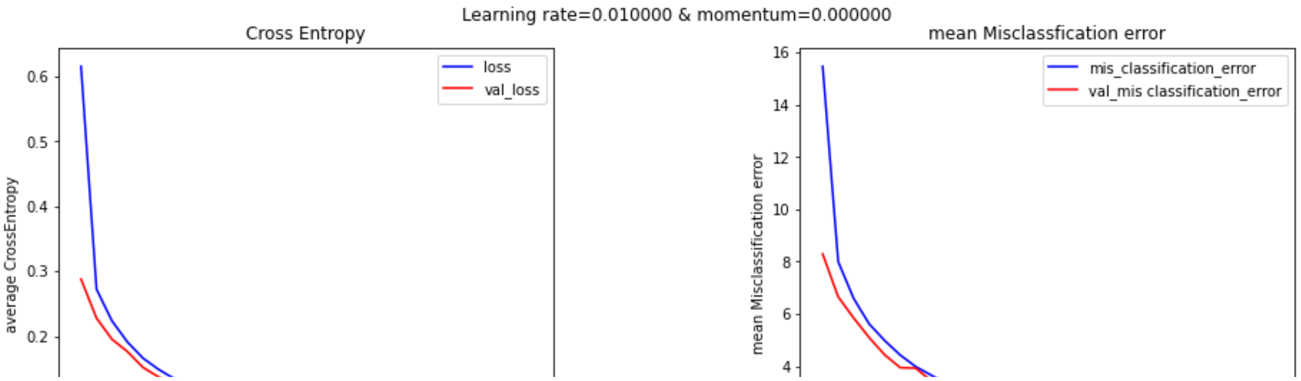
```
W = model.layers[0].get_weights()[0].reshape(3, 3, -1)
for i in range(W.shape[2]):
  plt.subplot(4, 8, i + 1)
  plt.imshow(W[:, :, i], cmap = 'gray')
  plt.xticks([])
  plt.yticks([])
plt.show()
```



4(d)

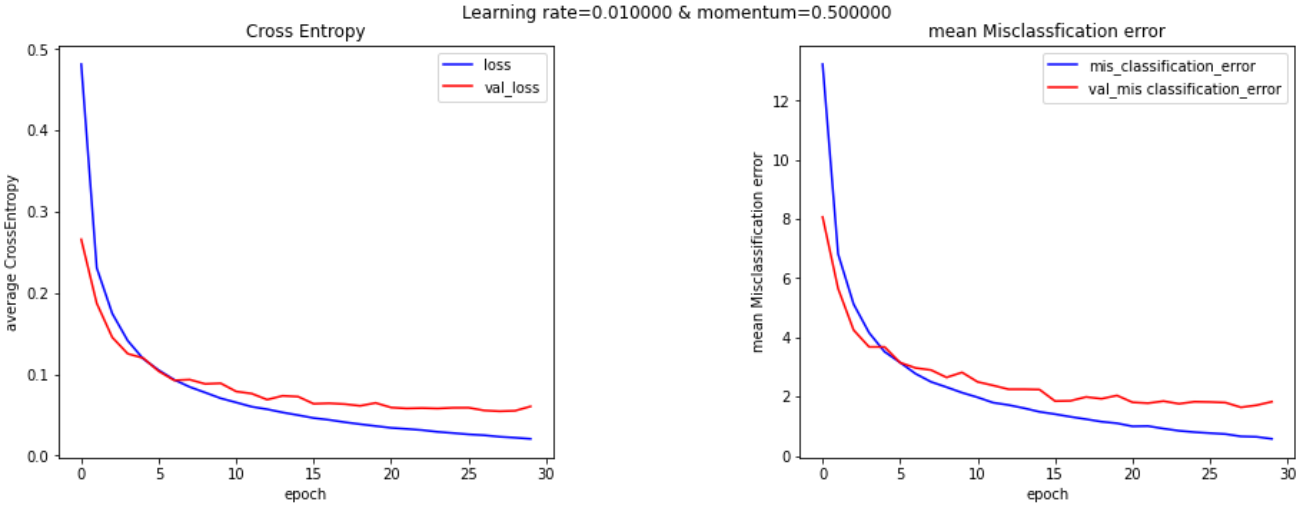
In [10]:

```
seed = 3

for lr in [0.01, 0.1, 0.2, 0.5]:
  for momentum in [0, 0.5, 0.9]:
    sgd = keras.optimizers.SGD(learning_rate = lr, momentum = momentum)
    np.random.seed(seed)

    #Create single layer NN
    model = Sequential()
    #Adding Hidden layer
    model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu', input_
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(units = 100, activation = 'relu'))
    model.add(Dense(units = 10, activation = 'softmax'))
    #Compiling the model
    model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCross
    history = model.fit(X_train_new1, Y_train, epochs = 30,  validation_data =

    #Plot
    f, ax = plt.subplots(1, 2, figsize = (15, 5))
    plt.subplots_adjust(left = 0.1, bottom = 0.1, right = 0.9, top = 0.9, wsp
    ax[0].plot(history.history['loss'], color = 'blue', label = "Training los
    ax[0].plot(history.history['val_loss'], color = 'red', label = "validatio
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('average CrossEntropy')
    ax[0].set_title('Cross Entropy')
    ax[0].legend(['loss', 'val_loss'], fontsize = 10)

    x = np.repeat(1, 30)
    ax[1].plot((x - history.history['accuracy'])*100, color = 'blue', label =
    ax[1].plot((x - history.history['val_accuracy'])*100, color = 'red', labe
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('mean Misclassification error')
    ax[1].set_title('mean Misclassfication error')
    ax[1].legend(['mis_classification_error', 'val_mis classification_error']

    f.suptitle('Learning rate=%2f & momentum=%2f'%(lr, momentum))
    plt.show()
    print("The misclassification error for test set is %3f"%(((x - history.hi
```

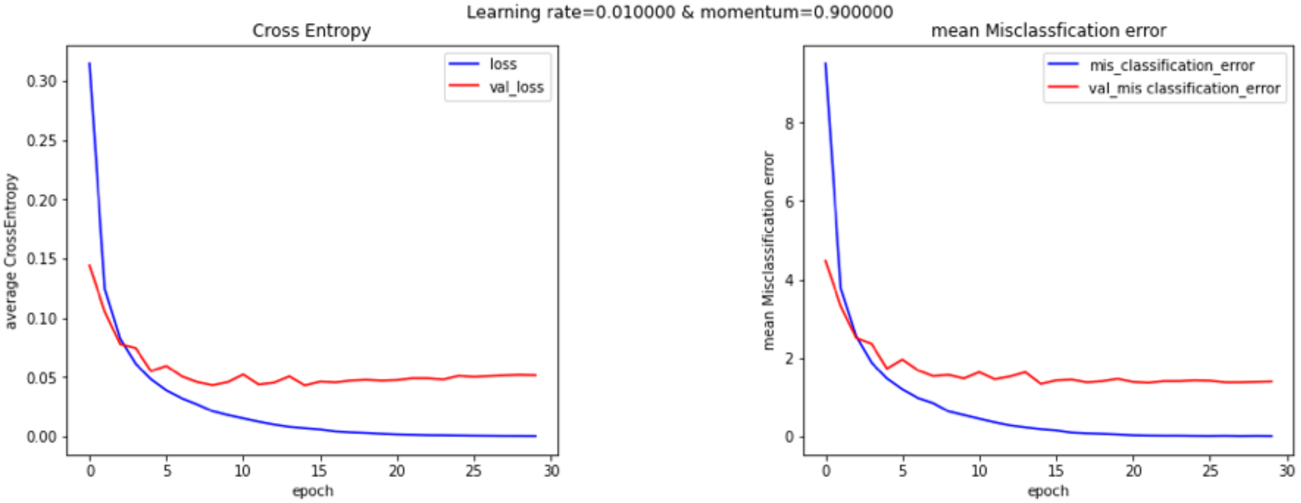


0.1
0 5 10 15 20 25 30
epoch
2
0 5 10 15 20 25 30
epoch

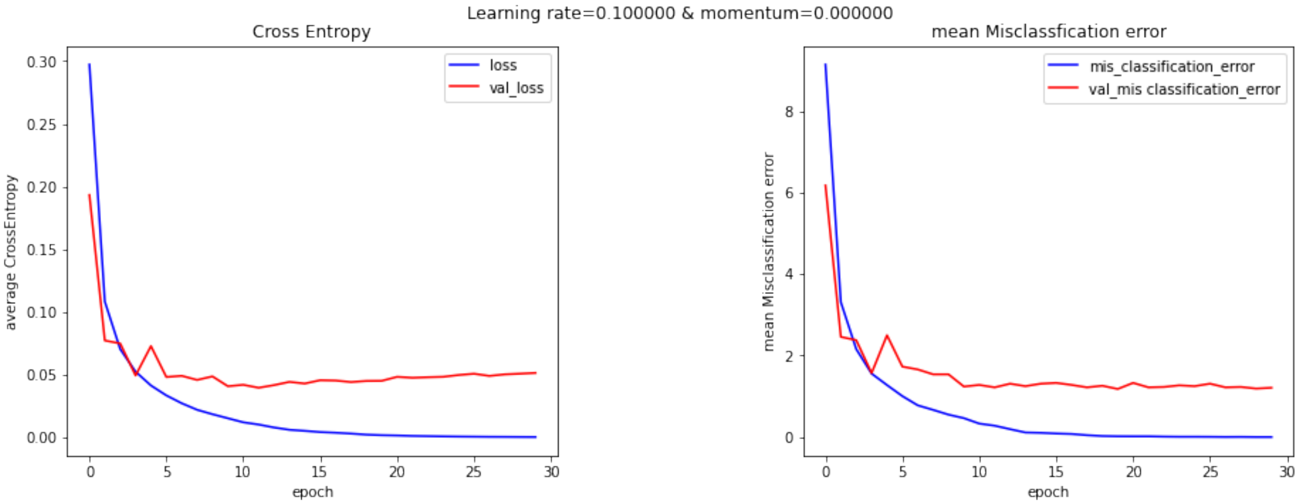
```
The misclassification error for test set is 3.259999%.
```


Learning rate=0.010000 & momentum=0.500000

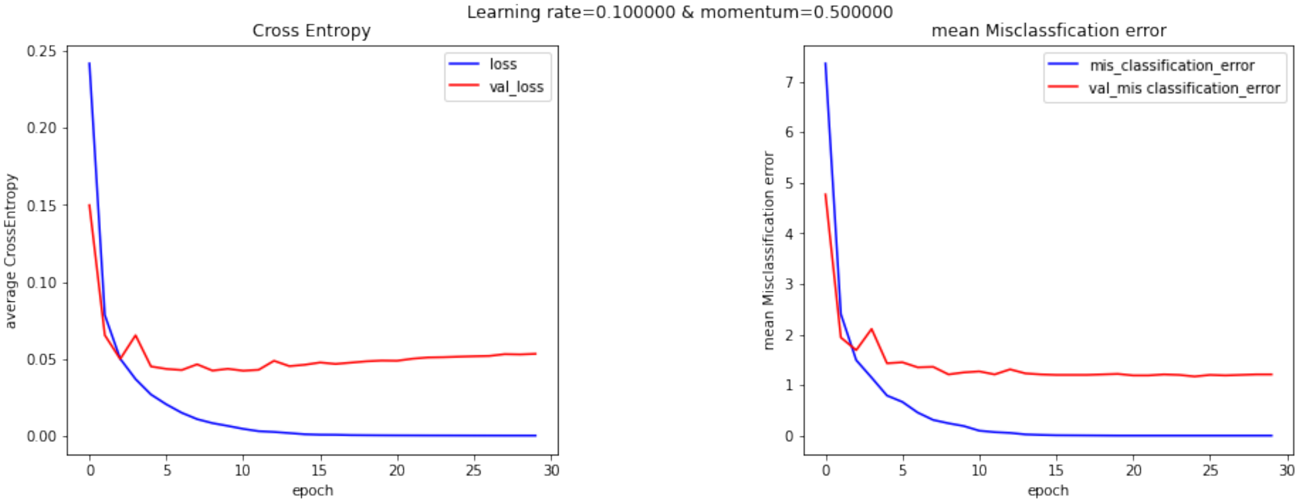
```
The misclassification error for test set is 2.819997%.
```


Learning rate=0.010000 & momentum=0.900000

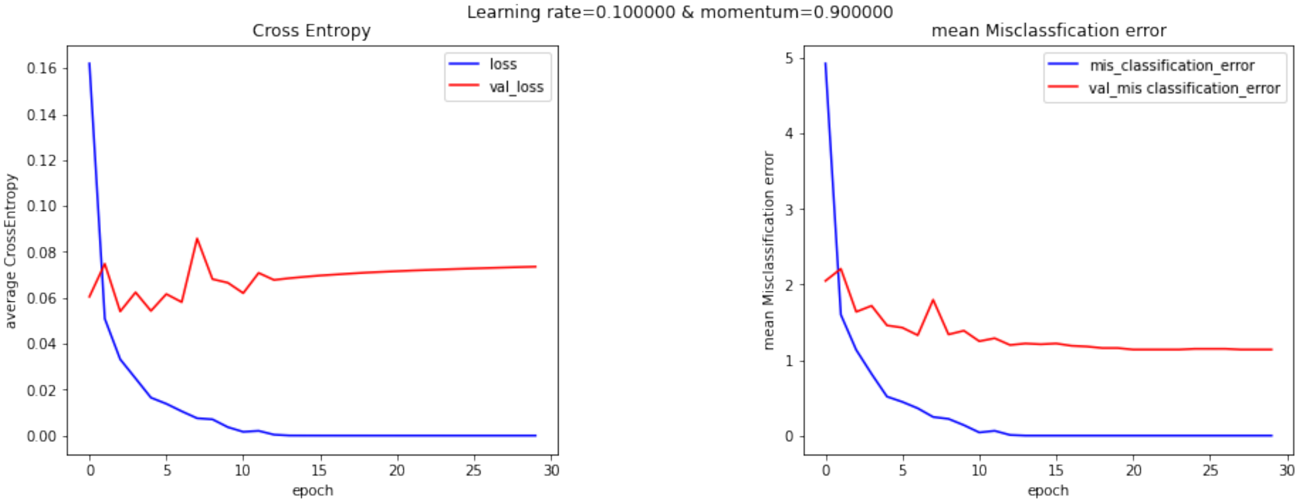
```
The misclassification error for test set is 1.470000%.
```


Learning rate=0.100000 & momentum=0.000000

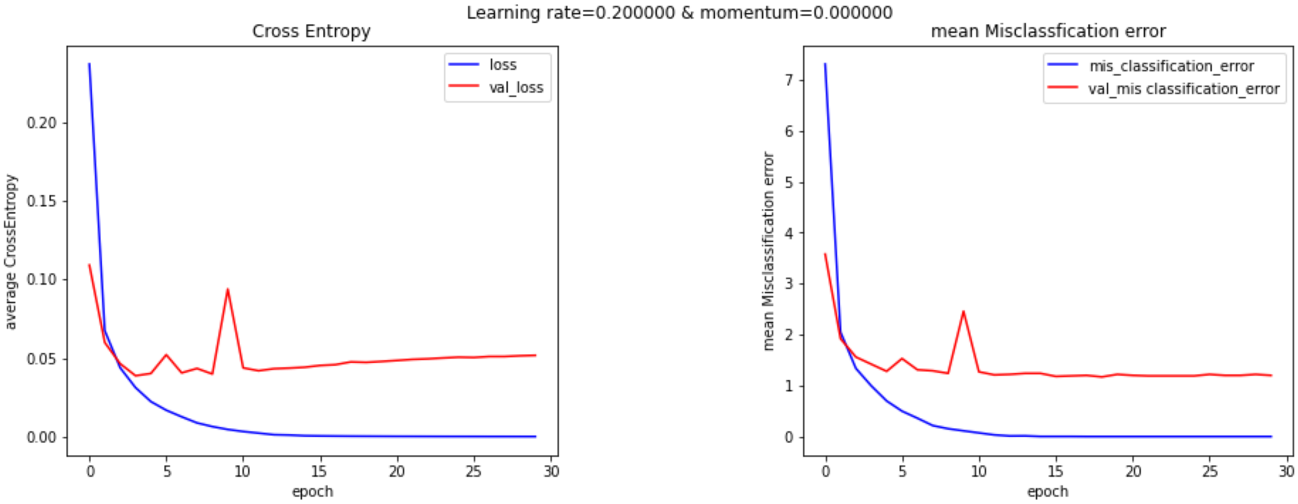
```
The misclassification error for test set is 1.239997%.
```



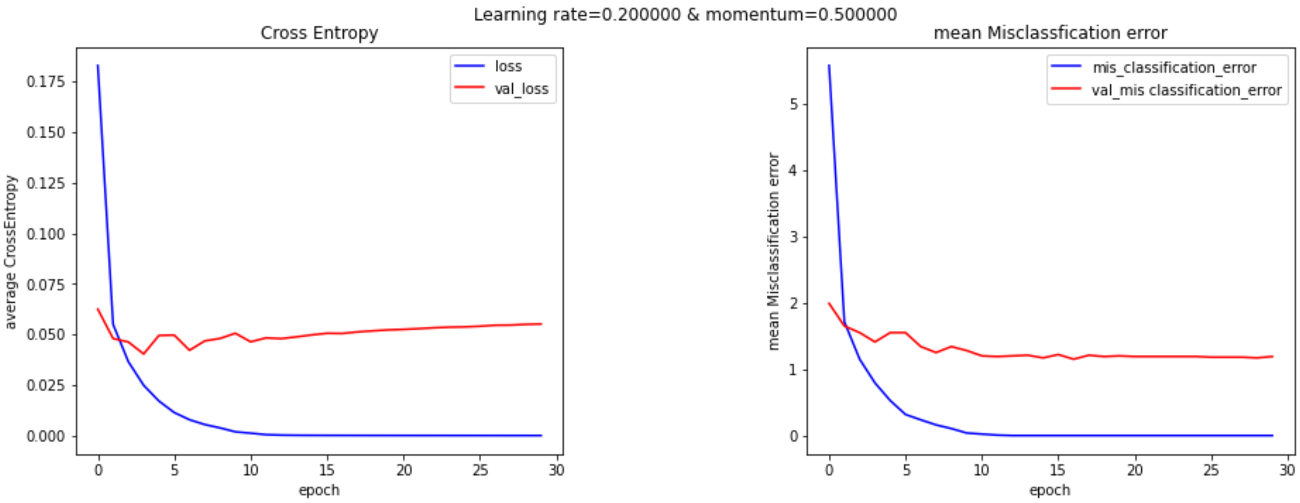
```
The misclassification error for test set is 1.249999%.
```



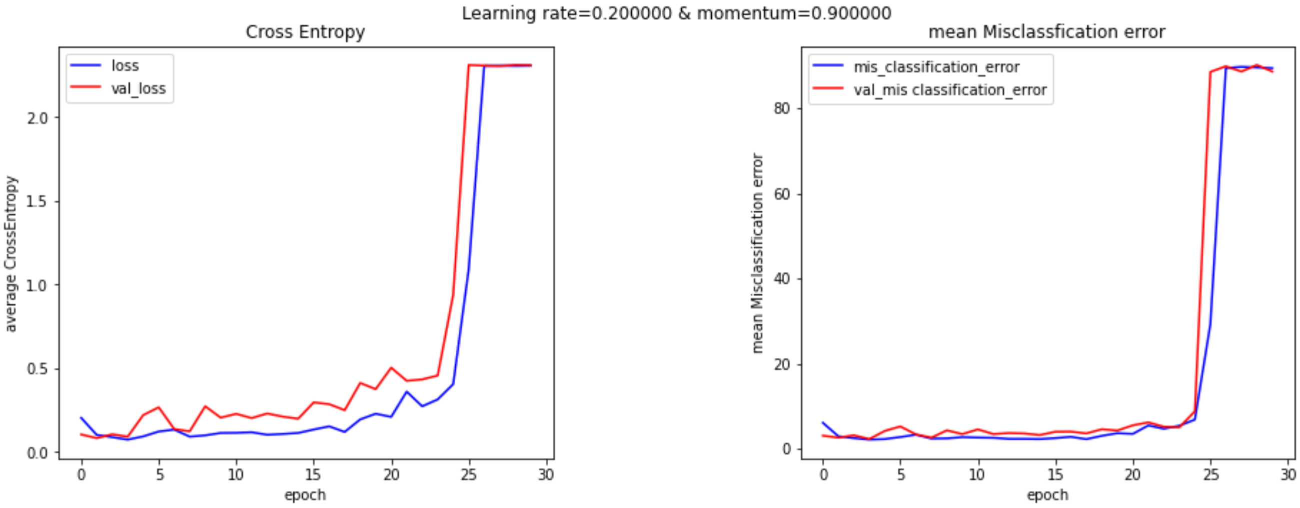
```
The misclassification error for test set is 1.389998%.
```



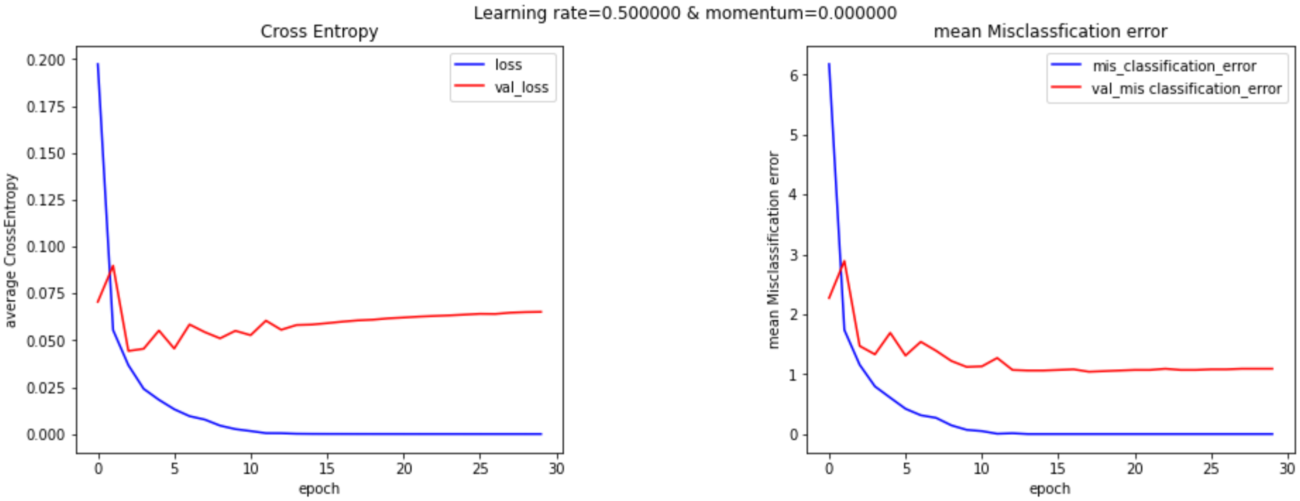
```
The misclassification error for test set is 2.460003%.
```



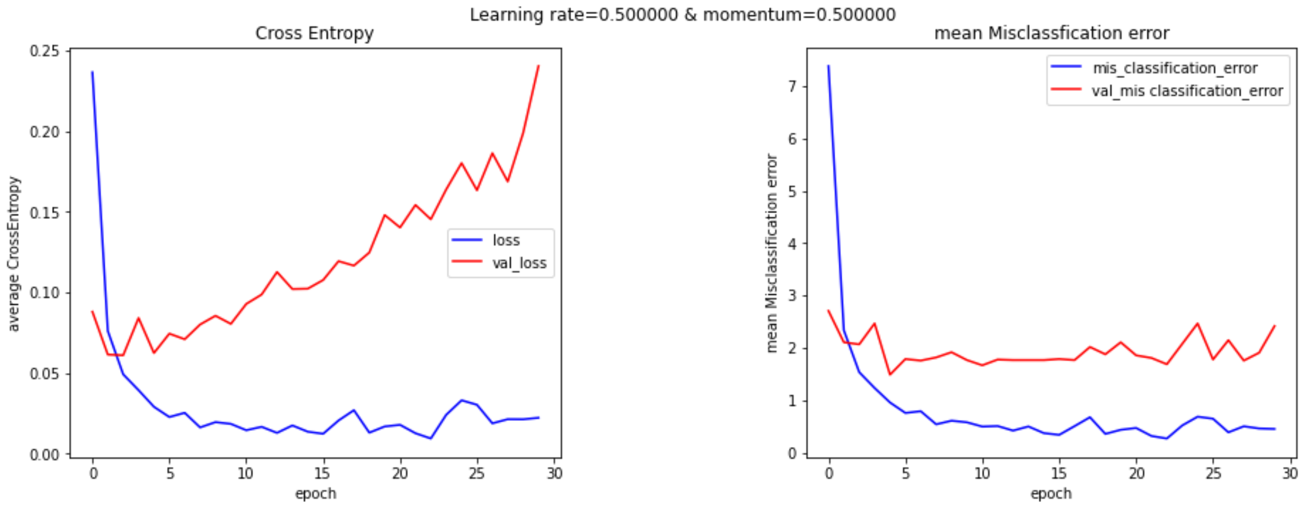
```
The misclassification error for test set is 1.279998%.
```



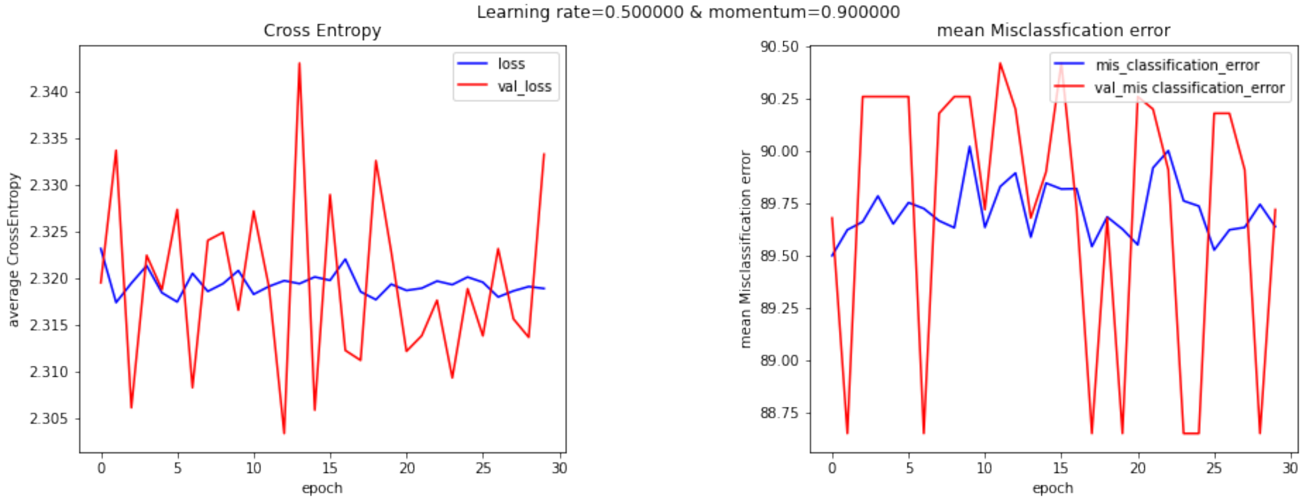
```
The misclassification error for test set is 3.420001%.
```



```
The misclassification error for test set is 1.120001%.
```



```
The misclassification error for test set is 1.770002%.
```



```
The misclassification error for test set is 90.260000%.
```

From above,we can see that most of the set are stable when epoch > 20.When learning rate = 0.5 or momentum = 0.9,it is less stable than other parameters.And when rate = 0.5 and momentum = 0.9,it is fluctuated.The best value of the parameters of the single layer is learning rate = 0.1 and momentum = 0.5, with the misclassificaiton error = 2.249998%, and for one layer of 2-D CNN, the misclassfication error = 1.120001%, it clearly shows that one layer of 2-D CNN is outstanding.For parameters visulization, in the last row,most dark parts are distributed on sides. And it is like the overall trend.

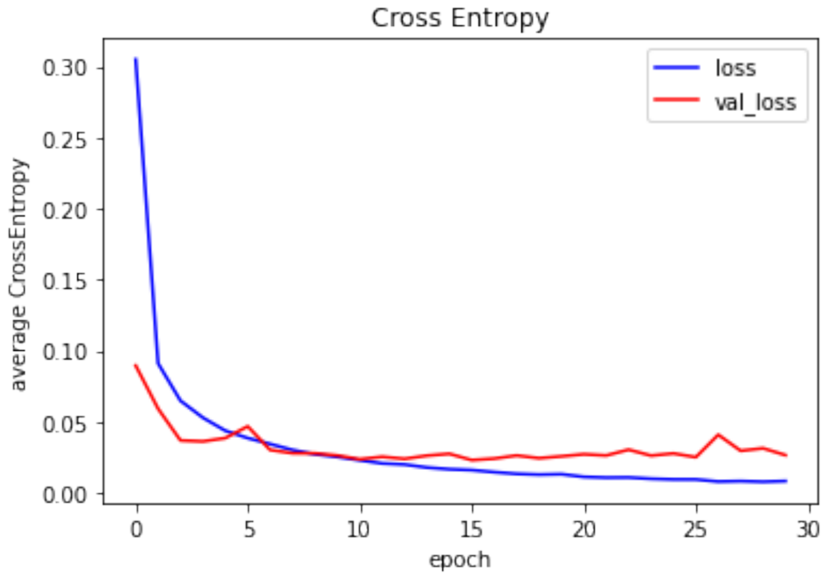
5(a)

In [15]:

```
from keras.layers import Dropout
```

In [15]:

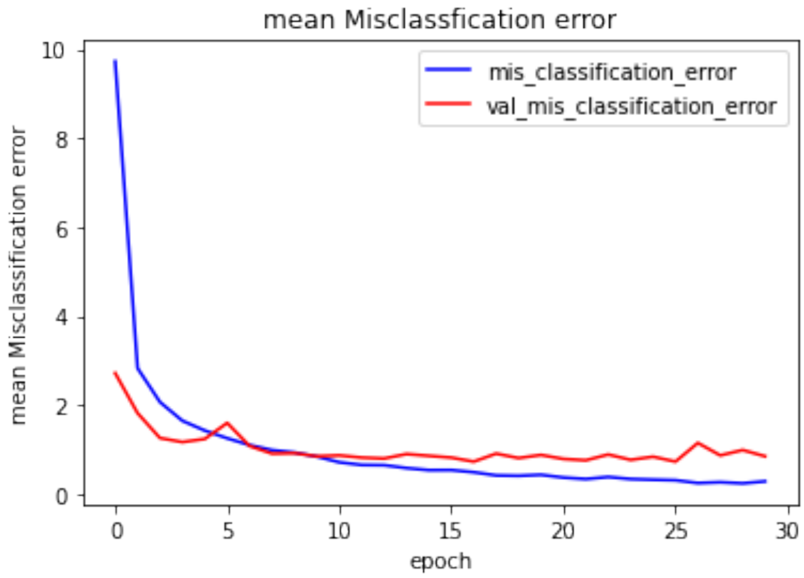
```
np.random.seed(3)
sgd = keras.optimizers.SGD(learning_rate = 0.1)
#Create single layer NN
model = Sequential()
#Adding Hidden layer
model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu', input_sha
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(64, (3, 3), padding = "same", activation = "relu"))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(units = 100, activation = 'relu'))
model.add(Dense(units = 10, activation = 'softmax'))
#Compiling the model
model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCrossentr
history = model.fit(X_train_new1, Y_train, epochs = 30, validation_data = (X_

#Plot
plt.plot(history.history['loss'], color = 'blue', label = "Training loss")
plt.plot(history.history['val_loss'], color = 'red', label = "validation loss
plt.xlabel('epoch')
plt.ylabel('average CrossEntropy')
plt.title('Cross Entropy')
plt.legend(['loss', 'val_loss'], fontsize = 10)

plt.show()
```



5（b)

In [16]:

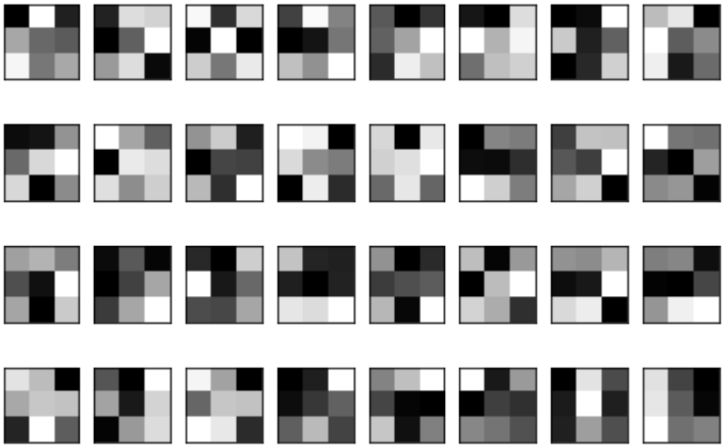
```
x = np.repeat(1, 30)
plt.plot((x - history.history['accuracy'])*100, color = 'blue', label = "mis_
plt.plot((x - history.history['val_accuracy'])*100, color = 'red', label = "v
plt.xlabel('epoch')
plt.ylabel('mean Misclassification error')
plt.title('mean Misclassfication error')
plt.legend(['mis_classification_error', 'val_mis_classification_error'], font
plt.show()
```



5(c)

In [19]:

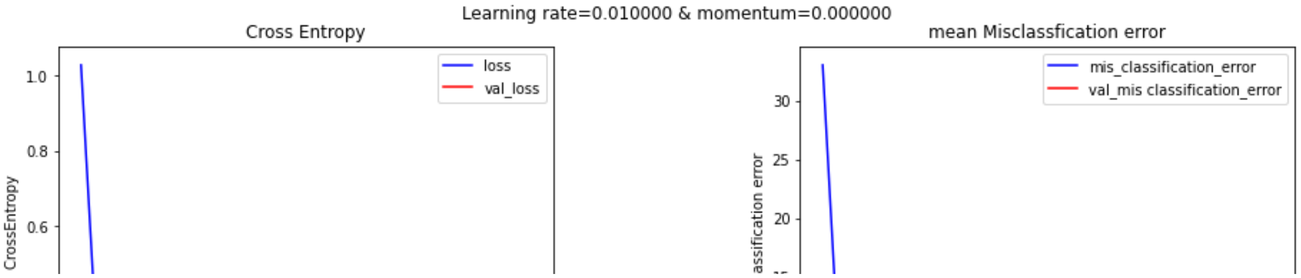
```
W = model.layers[0].get_weights()[0].reshape(3, 3, -1)
for i in range(W.shape[2]):
  plt.subplot(4, 8, i + 1)
  plt.imshow(W[:, :, i], cmap = 'gray')
  plt.xticks([])
  plt.yticks([])
plt.show()
```

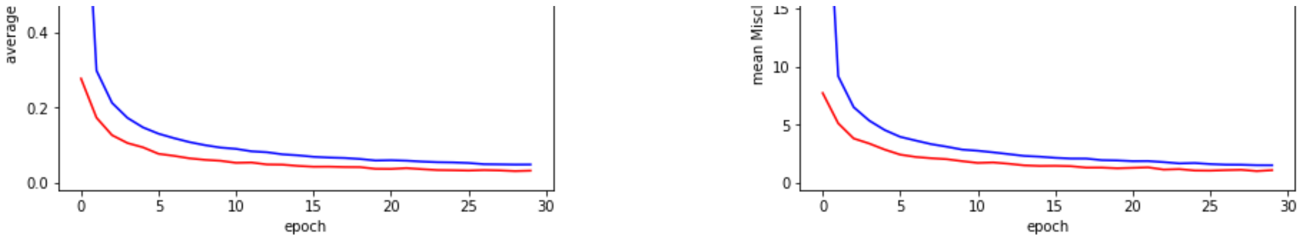


5(d)

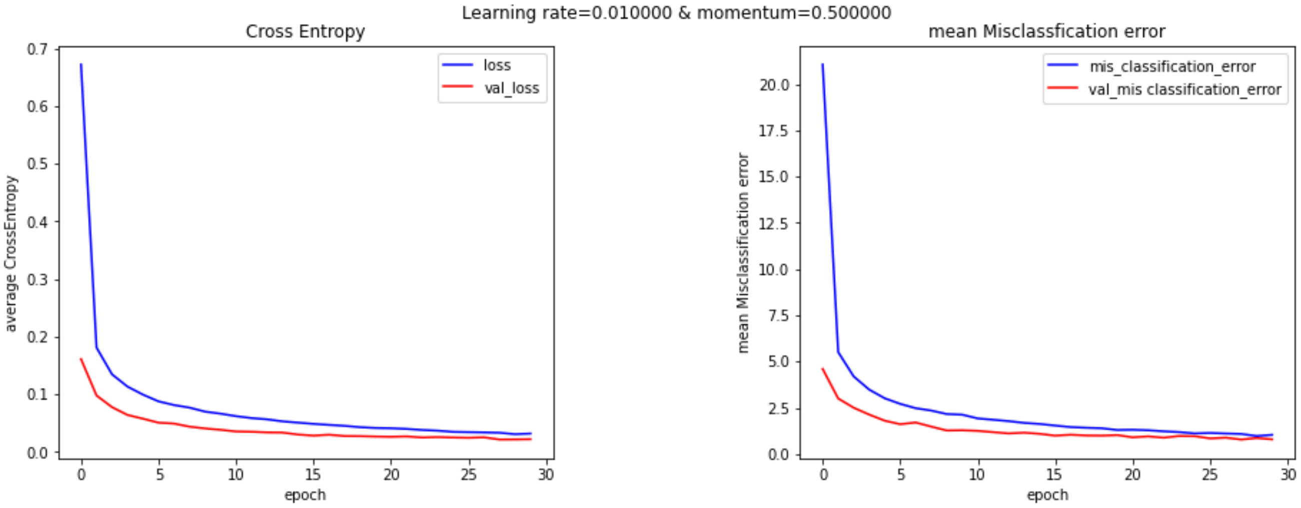
In [ ]:

```
seed = 3

for lr in [0.01, 0.1, 0.2, 0.5]:
  for momentum in [0, 0.5, 0.9]:
    sgd = keras.optimizers.SGD(learning_rate = lr, momentum = momentum)
    np.random.seed(seed)

    #Create single layer NN
    model = Sequential()
    #Adding Hidden layer
    model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu', input_
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), padding = "same", activation = "relu"))
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(units = 100, activation = 'relu'))
    model.add(Dense(units = 10, activation = 'softmax'))
    #Compiling the model
    model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCross
    history = model.fit(X_train_new1, Y_train, epochs = 30,  validation_data 

    #Plot
    f, ax = plt.subplots(1, 2, figsize = (15, 5))
    plt.subplots_adjust(left = 0.1, bottom = 0.1, right = 0.9, top = 0.9, wsp
    ax[0].plot(history.history['loss'], color = 'blue', label = "Training los
    ax[0].plot(history.history['val_loss'], color = 'red', label = "validatio
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('average CrossEntropy')
    ax[0].set_title('Cross Entropy')
    ax[0].legend(['loss', 'val_loss'], fontsize = 10)

    x = np.repeat(1, 30)
    ax[1].plot((x - history.history['accuracy'])*100, color = 'blue', label =
    ax[1].plot((x - history.history['val_accuracy'])*100, color = 'red', labe
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('mean Misclassification error')
    ax[1].set_title('mean Misclassfication error')
    ax[1].legend(['mis_classification_error', 'val_mis classification_error']

    f.suptitle('Learning rate=%2f & momentum=%2f'%(lr, momentum))
    plt.show()
    print("The misclassification error for test set is %3f"%(((x - history.hi
```





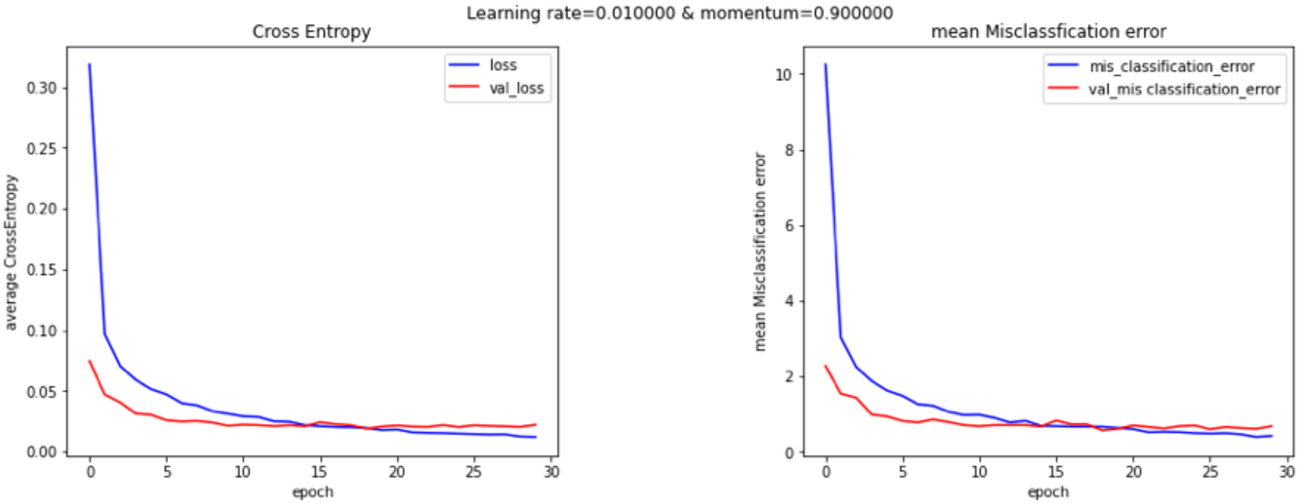
```
The misclassification error for test set is 1.889998%.
```


Learning rate=0.010000 & momentum=0.500000

Cross Entropy

mean Misclassfication error

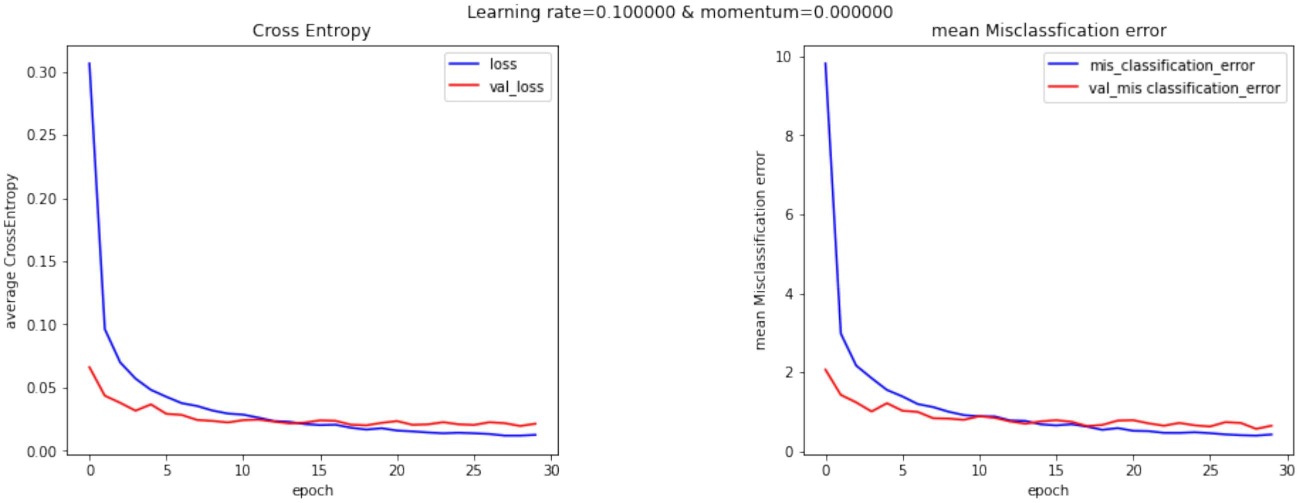
```
The misclassification error for test set is 1.279998%.
```


Learning rate=0.010000 & momentum=0.900000

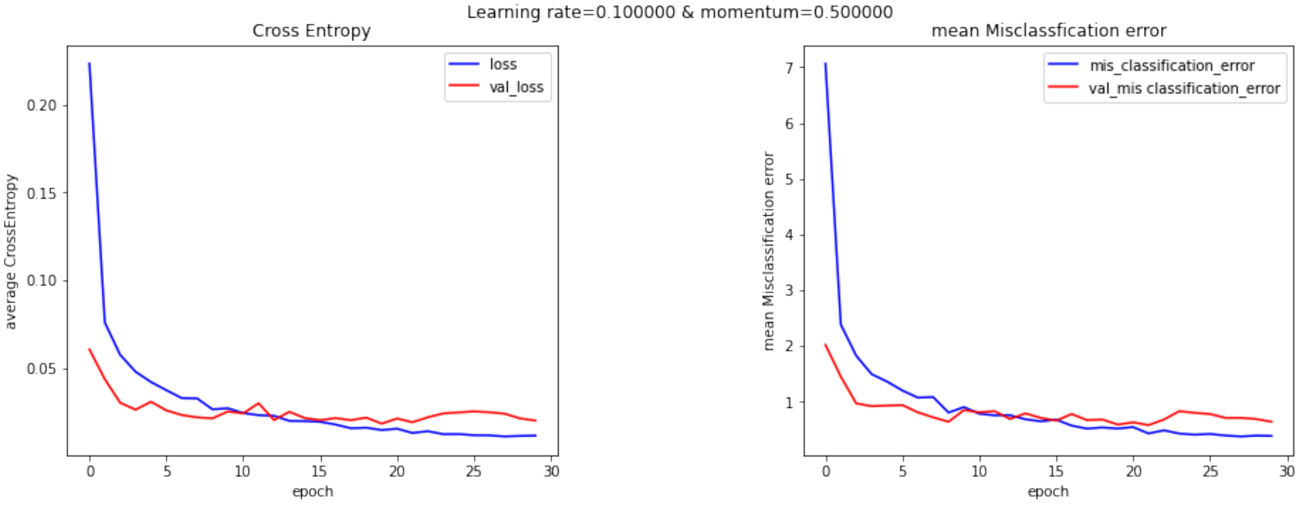
Cross Entropy

mean Misclassfication error

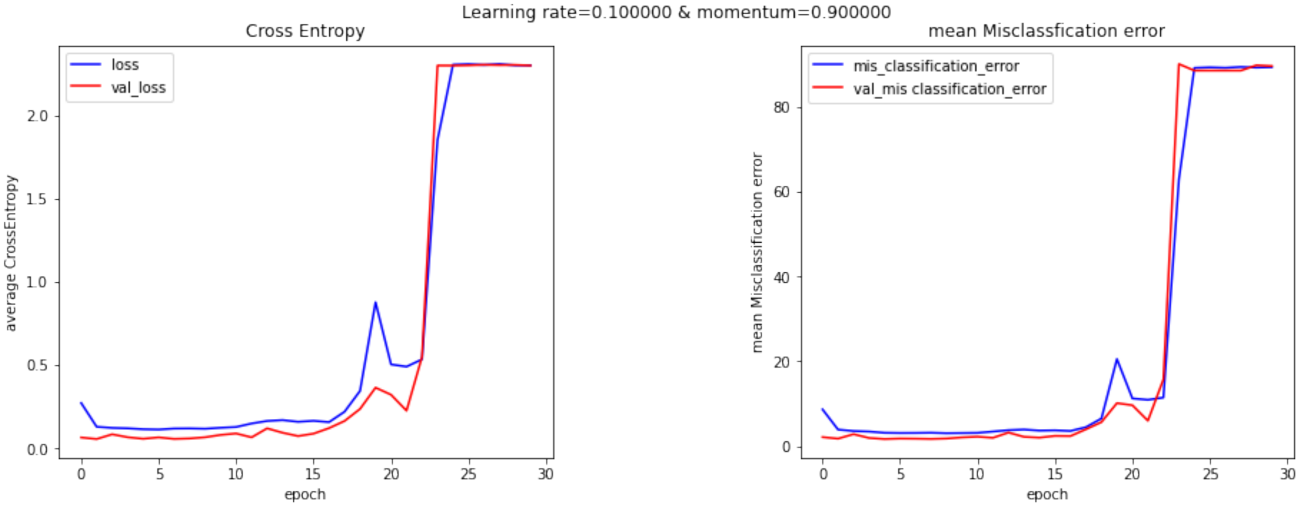
```
The misclassification error for test set is 0.720000%.
```



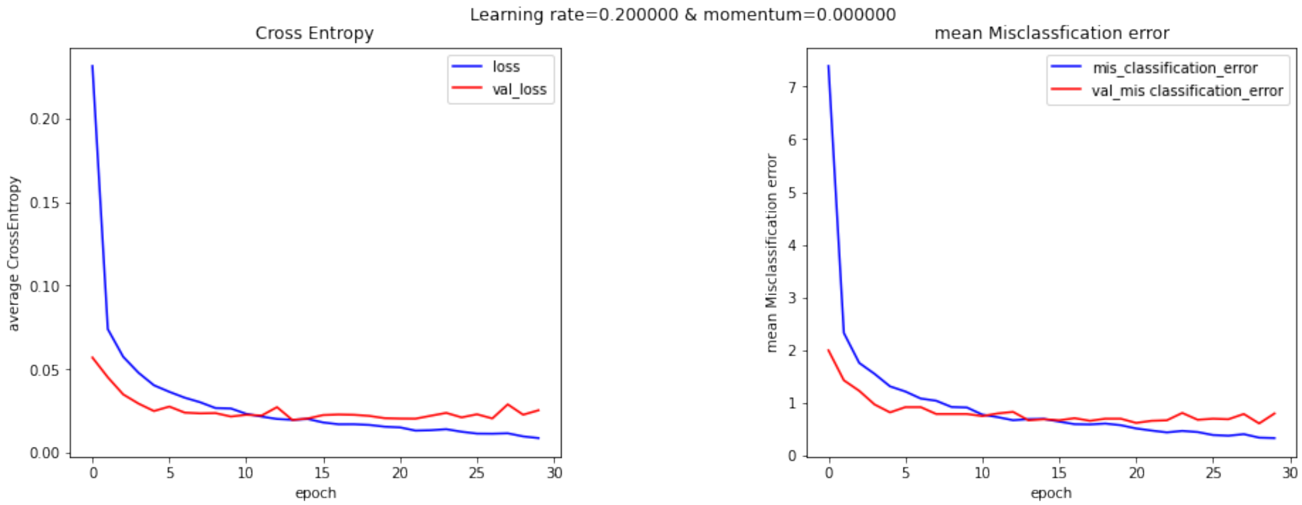
```
The misclassification error for test set is 0.800002%.
```



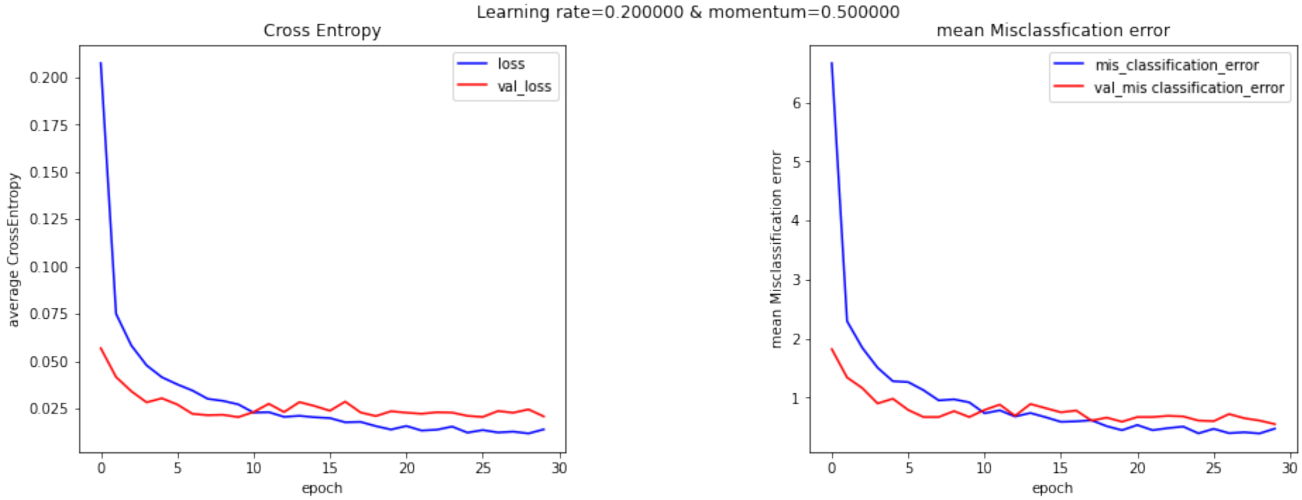
```
The misclassification error for test set is 0.849998%.
```



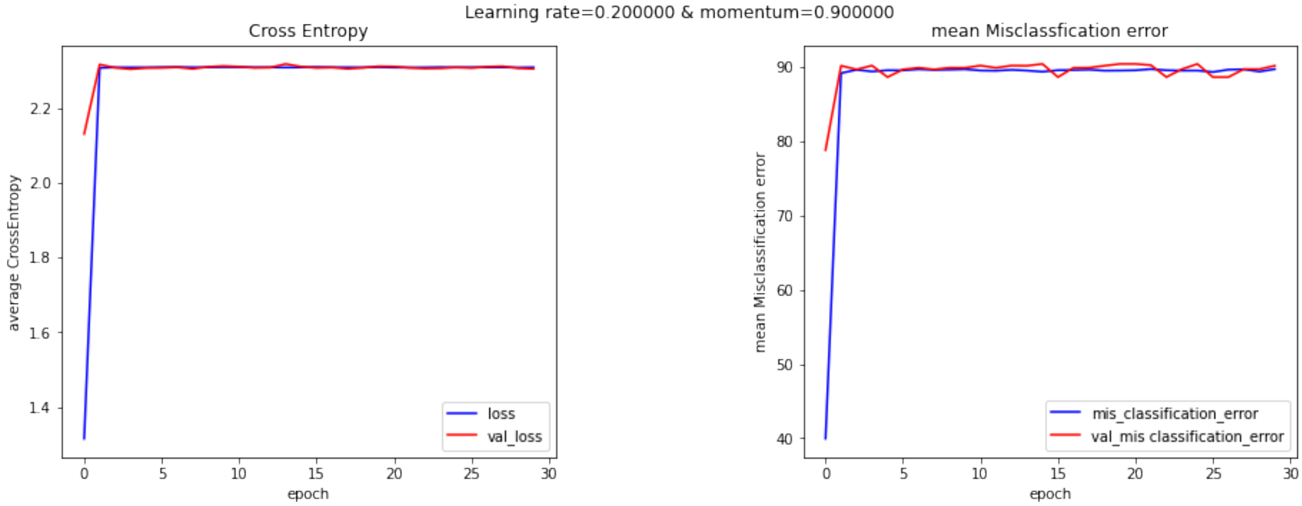
```
The misclassification error for test set is 2.029997%.
```



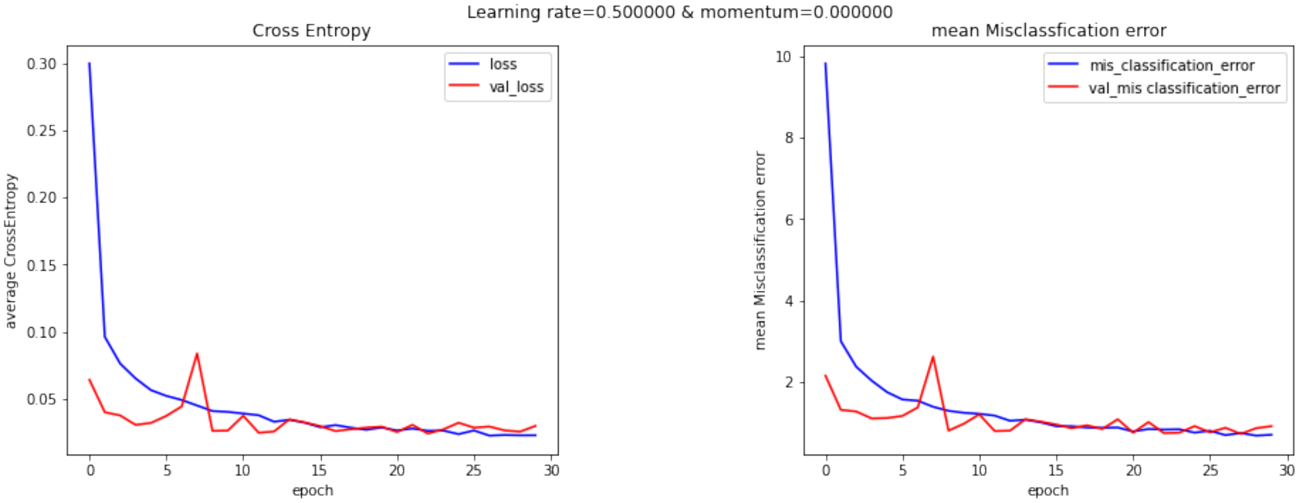
```
The misclassification error for test set is 0.779998%.
```



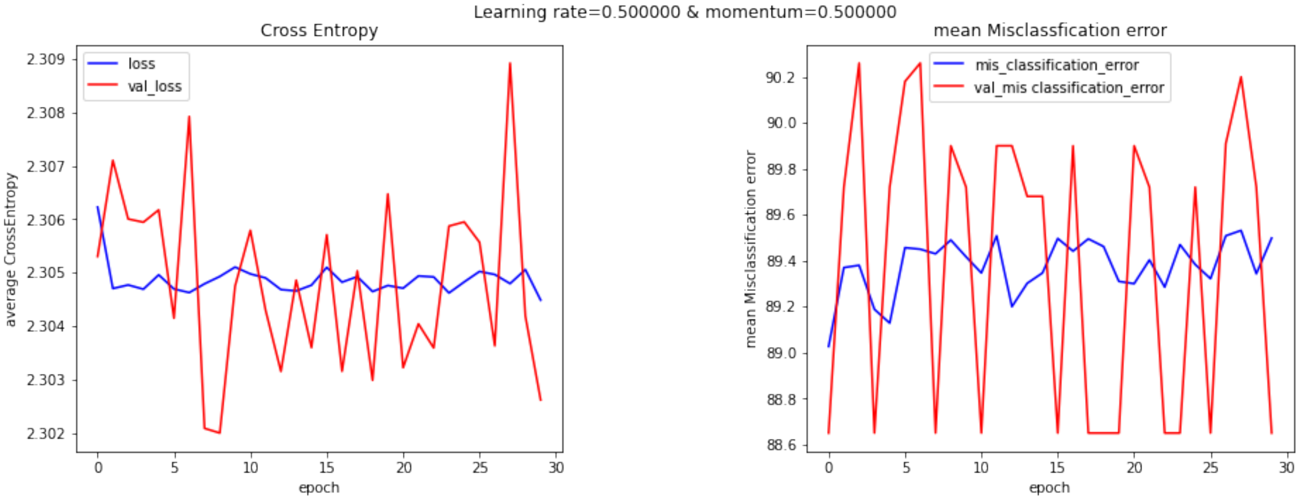
```
The misclassification error for test set is 0.669998%.
```



```
The misclassification error for test set is 89.900000%.
```



```
The misclassification error for test set is 0.980002%.
```



```
The misclassification error for test set is 89.720000%.
```

From the above, the error rate of most situations is smaller than one 2-D convolutional layer. The mean misclassificaiton error is more stable when learning rate = 0.01.And when momentum equals to 0.9,no matter which learning rate,it is unstble. The one layer of 2-D convolutional has the best value of the parameters when learning rate = 0.5, momentum = 0.0 with lower misclassification error, which is 1.120001% while the two layers of 2-D convolutional has the best value of the parameters when learning rate = 0.2, momentum = 0.5 with lower misclassification error, which is 0.669998%. Since the misclassification error = 0.669998% is smaller than the performance of SVM with Gaussian Kernel, our deep learning architecture beats SVM with Gaussian Kernel. For parameters visulization, in the third row, upper is dark, which is not too noisy, and too correlated.

6

In [21]:

```
# install pydrive to load data
!pip install -U -q PyDrive

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
import pandas as pd

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

In [25]:

```
id = "1Qt0RPvkQW8JgM_2lAhT0fy4G-H67Jivq"
file = drive.CreateFile({'id':id})
file.GetContentFile('test.txt')
test = pd.read_csv('test.txt',header=None).to_numpy()
```

In [26]:

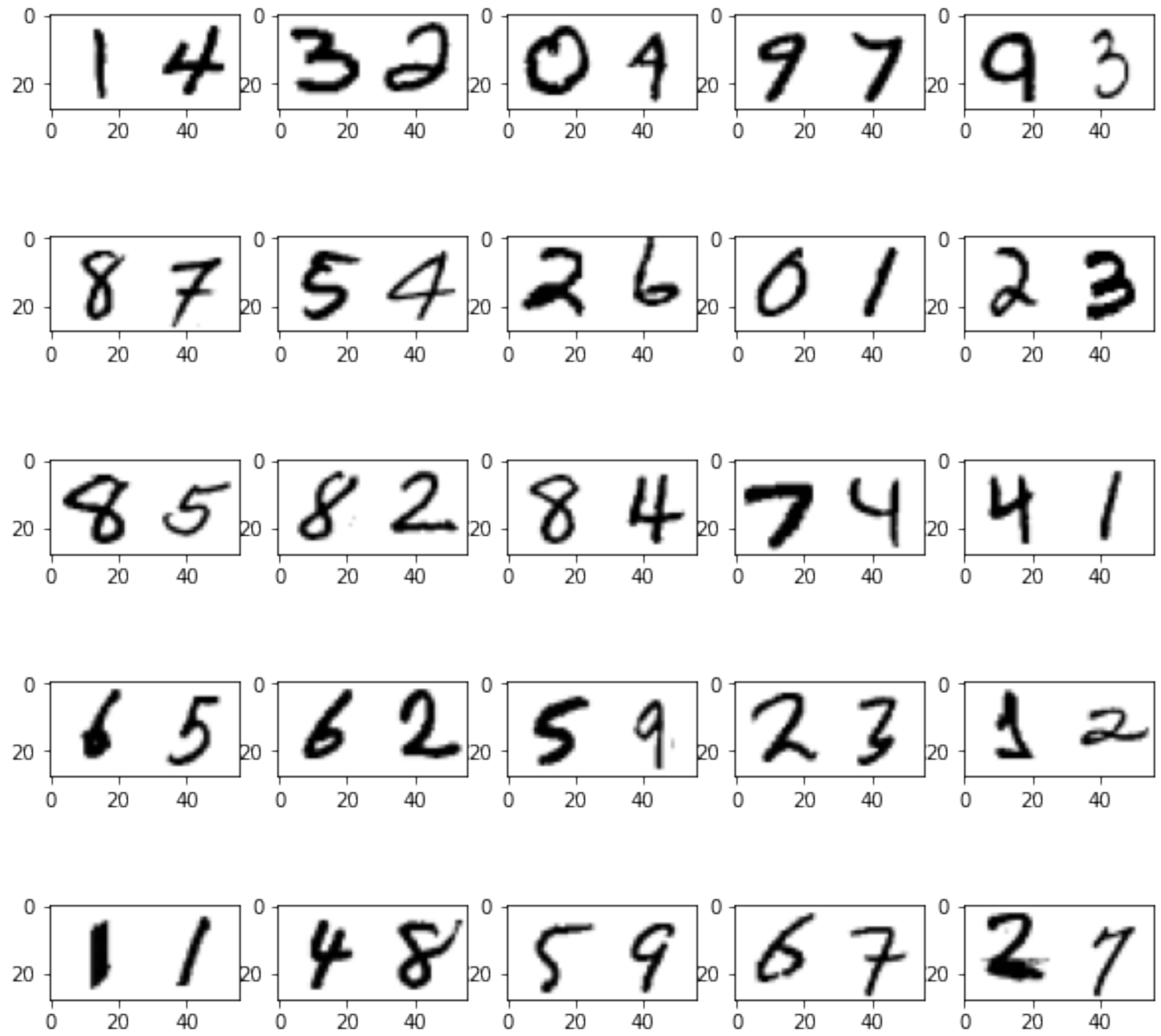
```
id = "17qlY1tu2DwYZ-komyT957yqHxhuqIgk1"
file = drive.CreateFile({'id':id})
file.GetContentFile('train.txt')
train = pd.read_csv('train.txt',header=None).to_numpy()
```

In [27]:

```
id = "18_OK5cP6bpLJ3yi_vhcFqrfwwN2Ts_Bs"
file = drive.CreateFile({'id':id})
file.GetContentFile('val.txt')
val = pd.read_csv('val.txt',header=None).to_numpy()
```

In [27]:

```
plt.figure(figsize=(10,10))
for i in range(25):
    s=train[i,:1568].reshape(28,56)
    plt.subplot(5,5,i+1)
    plt.grid(False)
    plt.imshow(s, cmap=plt.cm.binary)
plt.show()
```



In [28]:

```
print(train[0:25,-1].reshape(5,5))
```

```
[[ 5.  5.  4. 16. 12.]
 [15.  9.  8.  1.  5.]
 [13. 10. 12. 11.  5.]
 [11.  8. 14.  5.  3.]
 [ 2. 12. 14. 13.  9.]]
```

The pixels were scanned out in row-major and the relationship between the 2 digits and the last coordinate of each line is that the sum of the 2 digits is equal to the value of the last coordinate of each line

7

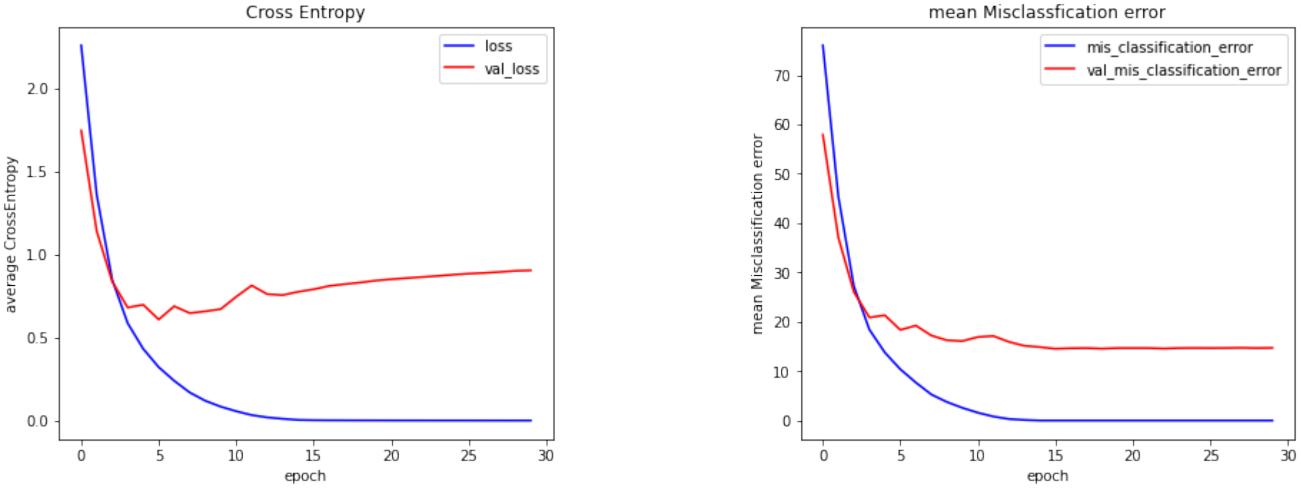
first:

In [28]:

```
X_train = train[:, :1568].reshape((-1, 28, 56, 1))
Y_train = train[:, -1]
X_val = val[:, :1568].reshape((-1, 28, 56, 1))
Y_val = val[:, -1]
X_test = test[:, :1568].reshape((-1, 28, 56, 1))
Y_test = test[:, -1]
```

In [19]:

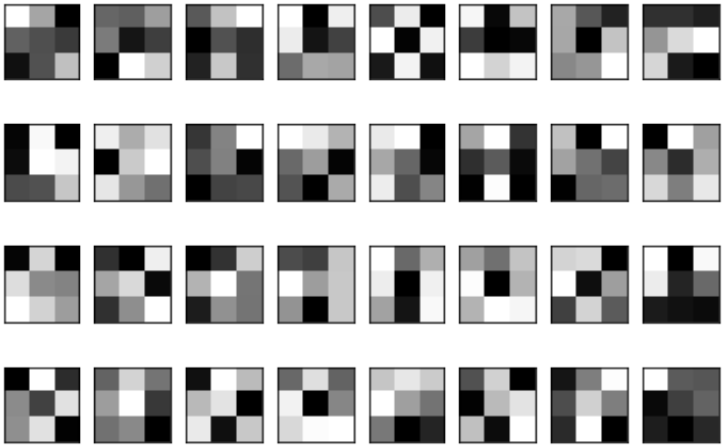
```
np.random.seed(3)
sgd = keras.optimizers.SGD(learning_rate = 0.1, momentum = 0.5)
#Create single layer NN
model = Sequential()
#Adding Hidden layers
model.add(Conv2D(32, (3, 3), padding = 'same', activation='relu', input_shape
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dense(units = 19))
#Compiling the model
model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCrossentr
history = model.fit(X_train, Y_train, epochs = 30, validation_data = (X_val, 

#Plot
f, ax = plt.subplots(1, 2, figsize = (15, 5))
plt.subplots_adjust(left = 0.1, bottom = 0.1, right = 0.9, top = 0.9, wspace 
ax[0].plot(history.history['loss'], color = 'blue', label = "Training loss")
ax[0].plot(history.history['val_loss'], color = 'red', label = "validation lo
ax[0].set_xlabel('epoch')
ax[0].set_ylabel('average CrossEntropy')
ax[0].set_title('Cross Entropy')
ax[0].legend(['loss','val_loss'], fontsize = 10)

x = np.repeat(1, 30)
ax[1].plot((x - history.history['accuracy'])*100, color = 'blue', label = "mi
ax[1].plot((x - history.history['val_accuracy'])*100, color = 'red', label = 
ax[1].set_xlabel('epoch')
ax[1].set_ylabel('mean Misclassification error')
ax[1].set_title('mean Misclassfication error')
ax[1].legend(['mis_classification_error', 'val_mis_classification_error'], fo

plt.show()

pred = np.argmax(model.predict(X_test), axis=1)
err = np.mean(pred != Y_test)*100
print("The misclassification error for validation set and test set are %3f"%(
      'and %3f'%(err)+'%.')
```



```
The misclassification error for validation set and test set are 16.100001% and
13.180000%.
```

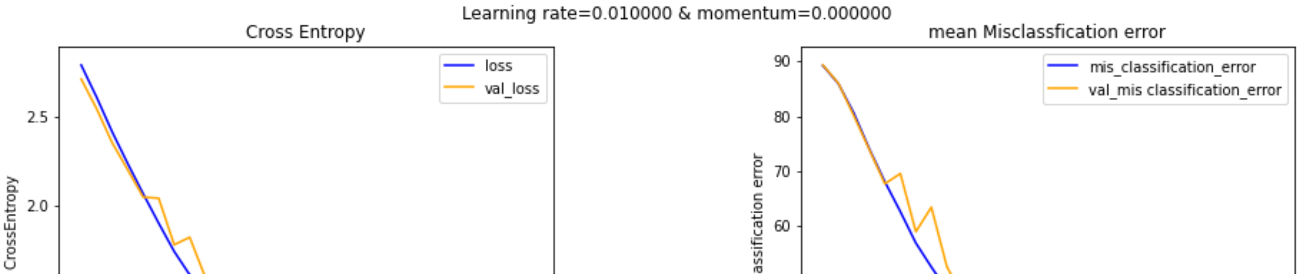
In [30]:

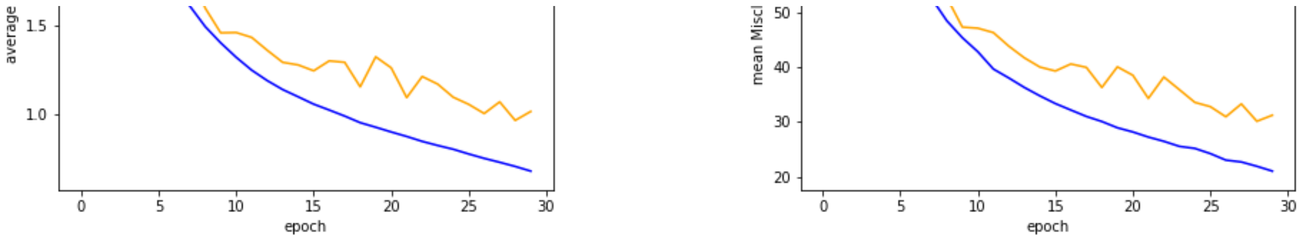
```
W = model.layers[0].get_weights()[0].reshape(3, 3, -1)
for i in range(W.shape[2]):
  plt.subplot(4, 8, i + 1)
  plt.imshow(W[:, :, i], cmap = 'gray')
  plt.xticks([])
  plt.yticks([])
plt.show()
```



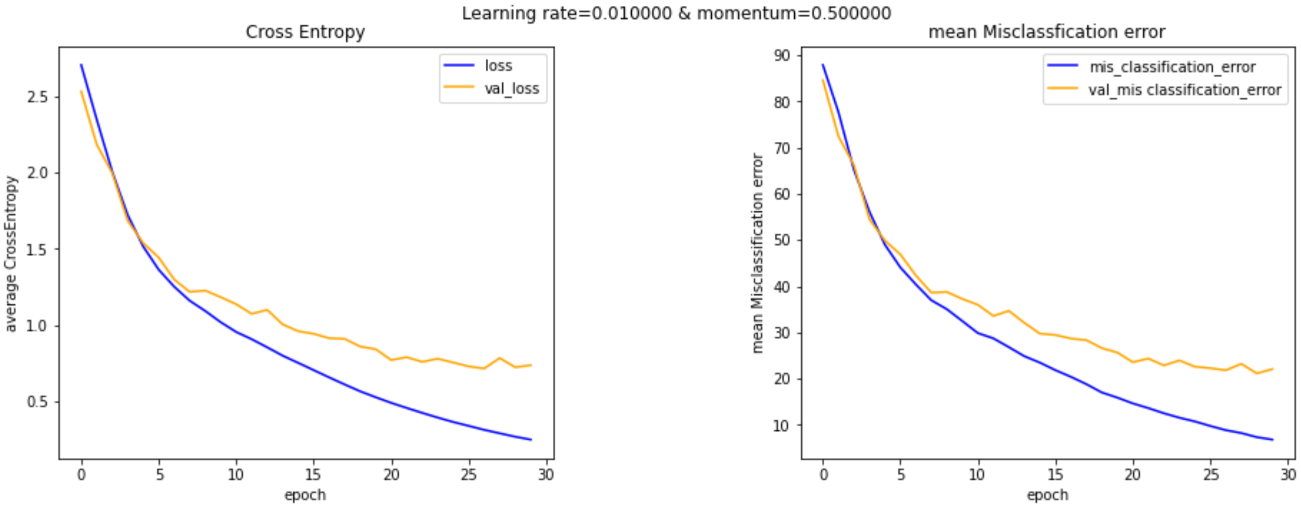
In [21]:

```
seed = 3

for lr in [0.01, 0.1, 0.2, 0.5]:
  for momentum in [0, 0.5, 0.9]:
    sgd = keras.optimizers.SGD(learning_rate = lr, momentum = momentum)
    np.random.seed(seed)

    #Create single layer NN
    model = Sequential()
    #Adding Hidden layer
    model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu', input_
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(units = 64, activation = 'relu'))
    model.add(Dense(units = 19))
    #Compiling the model
    model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCross
    history = model.fit(X_train, Y_train, epochs = 30,  validation_data = (X_

    #Plot
    f, ax = plt.subplots(1, 2, figsize = (15, 5))
    plt.subplots_adjust(left = 0.1, bottom = 0.1, right = 0.9, top = 0.9, wsp
    ax[0].plot(history.history['loss'], color = 'blue', label = "Training los
    ax[0].plot(history.history['val_loss'], color = 'orange', label = "valida
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('average CrossEntropy')
    ax[0].set_title('Cross Entropy')
    ax[0].legend(['loss', 'val_loss'], fontsize = 10)

    x = np.repeat(1, 30)
    ax[1].plot((x - history.history['accuracy'])*100, color = 'blue', label =
    ax[1].plot((x - history.history['val_accuracy'])*100, color = 'orange', l
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('mean Misclassification error')
    ax[1].set_title('mean Misclassfication error')
    ax[1].legend(['mis_classification_error', 'val_mis classification_error']

    f.suptitle('Learning rate=%2f & momentum=%2f'%(lr, momentum))
    plt.show()
    pred = np.argmax(model.predict(X_test), axis=1)
    err = np.mean(pred != Y_test)*100
    print("The misclassification error for validation set and test set are %3
          'and %3f'%(err)+'%.')
```

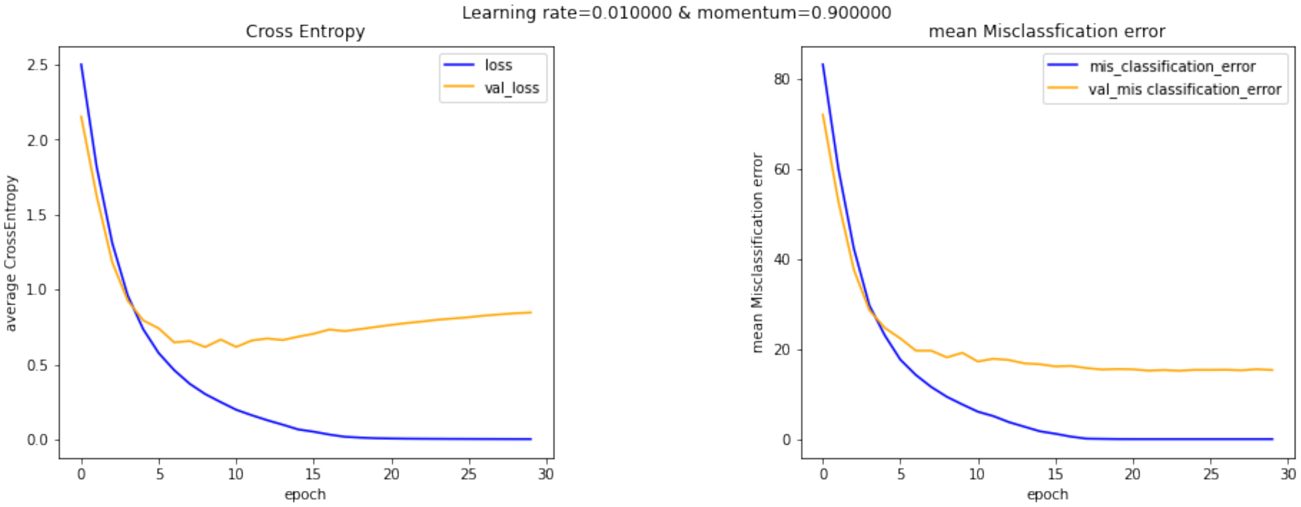




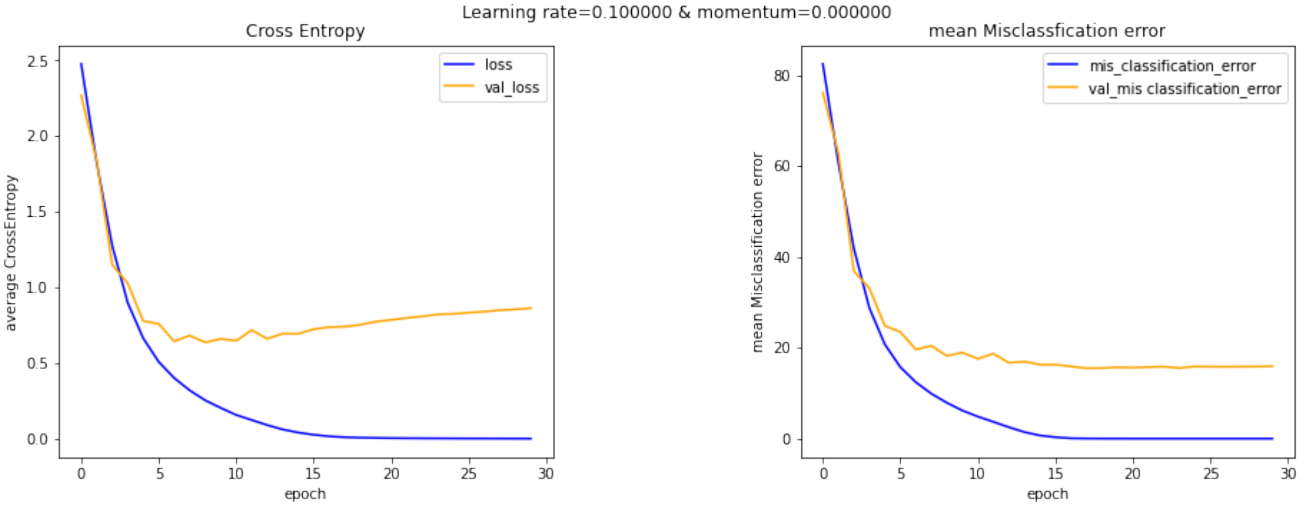
```
The misclassification error for validation set and test set are 47.259998% and
29.560000%.
```



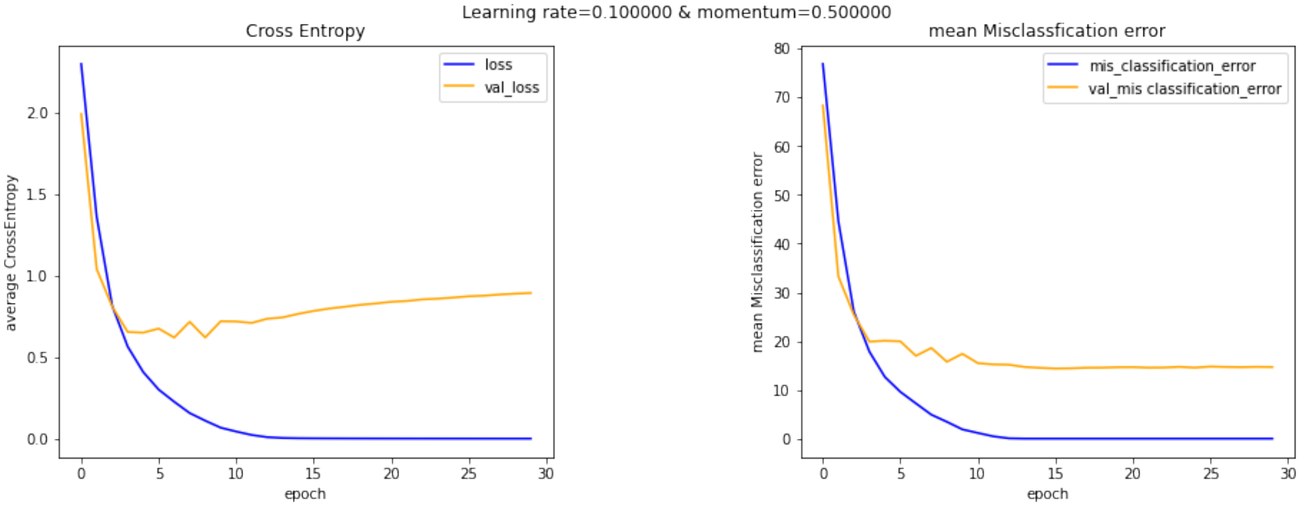
```
The misclassification error for validation set and test set are 37.260002% and
19.380000%.
```



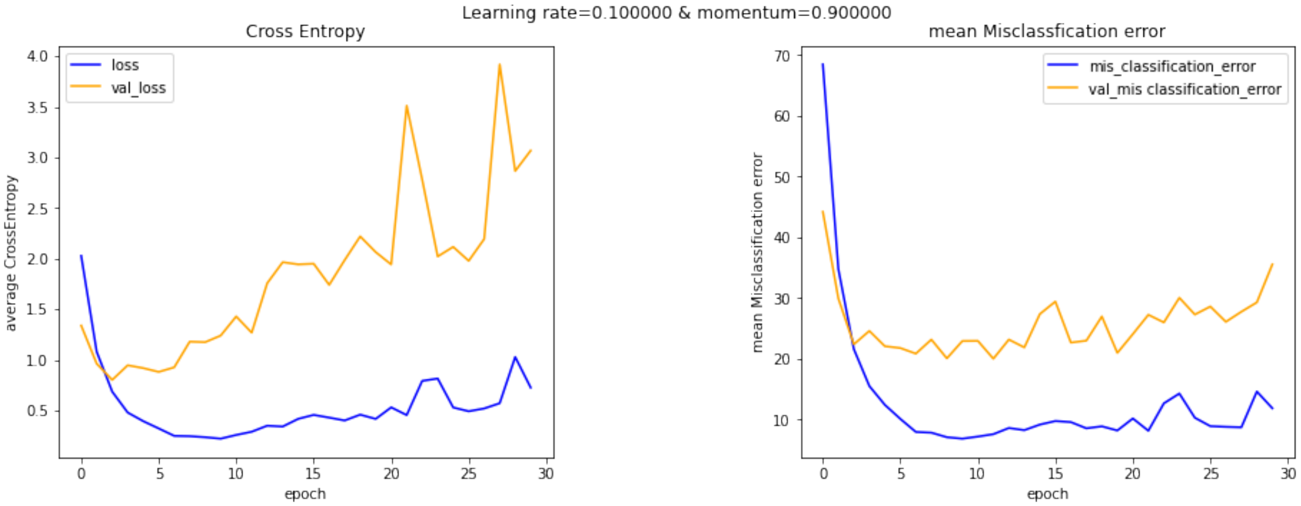
```
The misclassification error for validation set and test set are 19.139999% and
14.820000%.
```



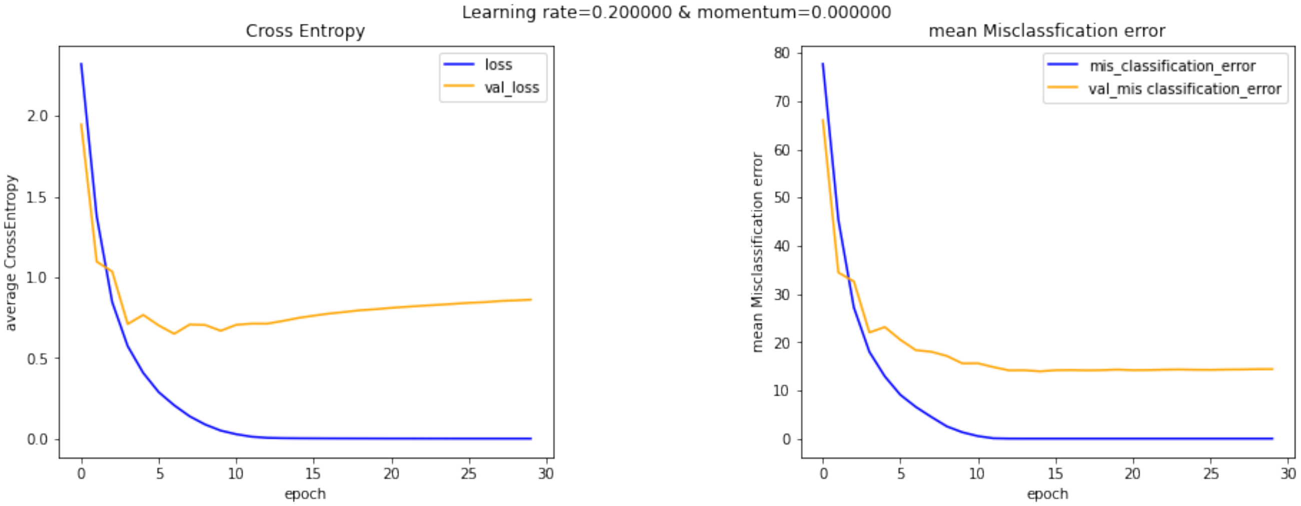
```
The misclassification error for validation set and test set are 18.919998% and
14.140000%.
```



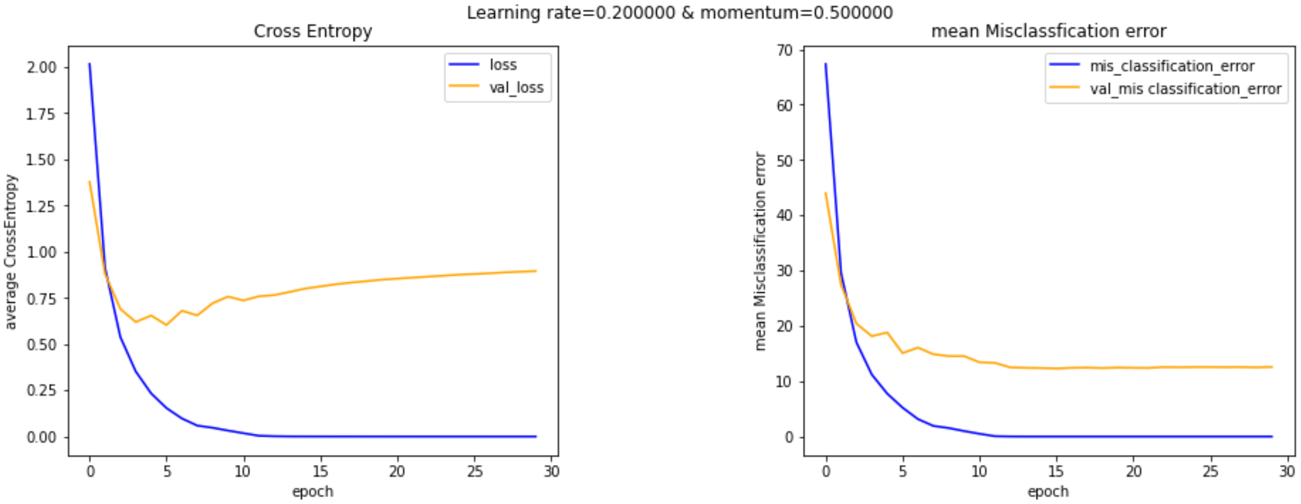
```
The misclassification error for validation set and test set are 17.400002% and
13.560000%.
```



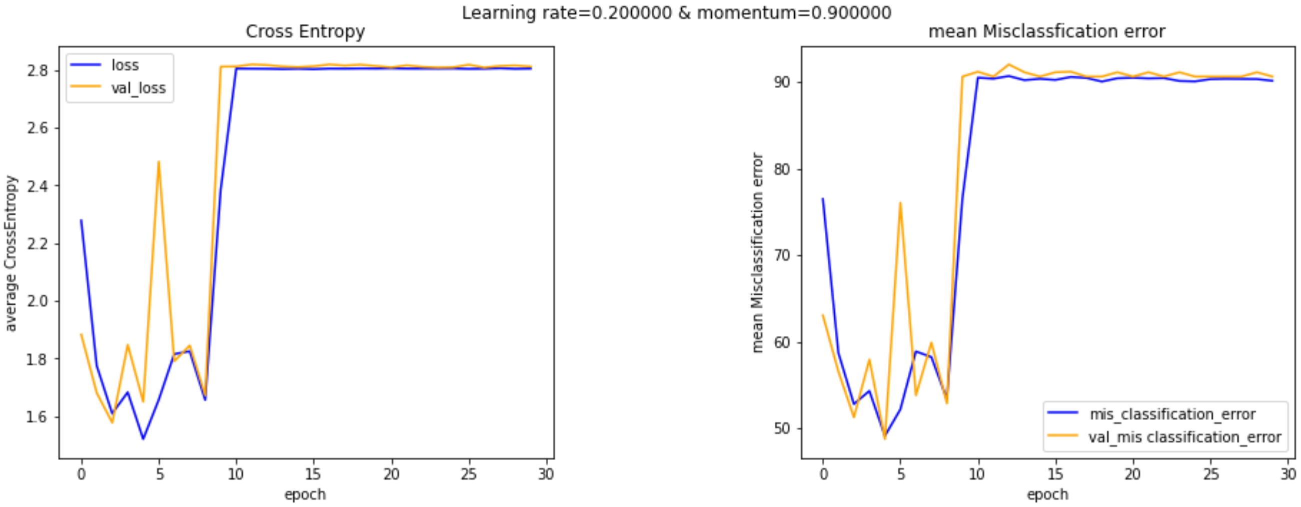
```
The misclassification error for validation set and test set are 22.939998% and
32.920000%.
```



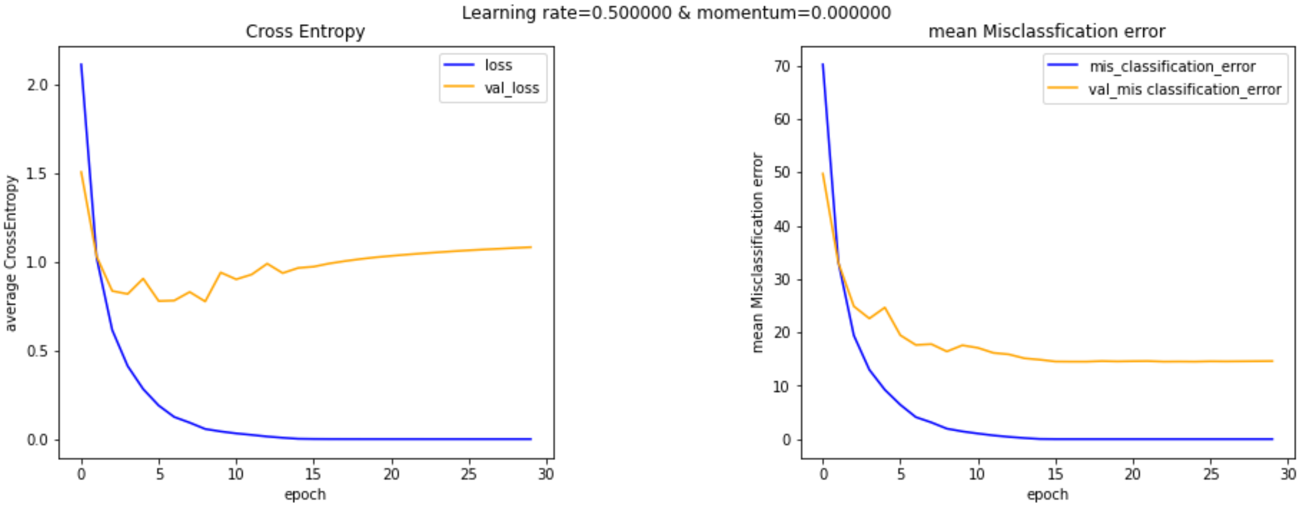
```
The misclassification error for validation set and test set are 15.600002% and
12.840000%.
```



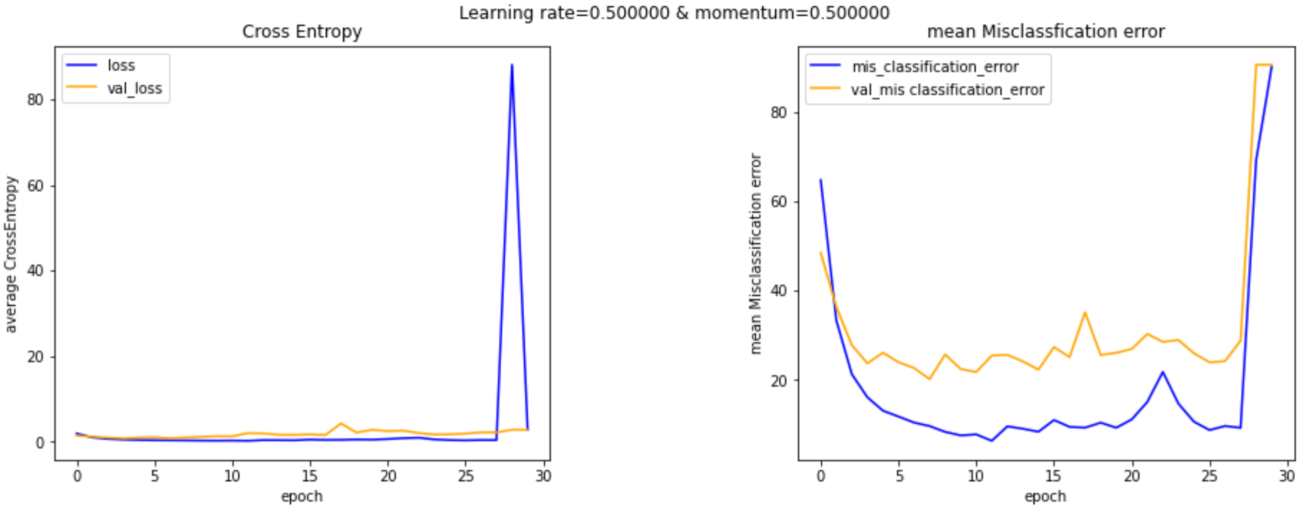
```
The misclassification error for validation set and test set are 14.539999% and
11.360000%.
```



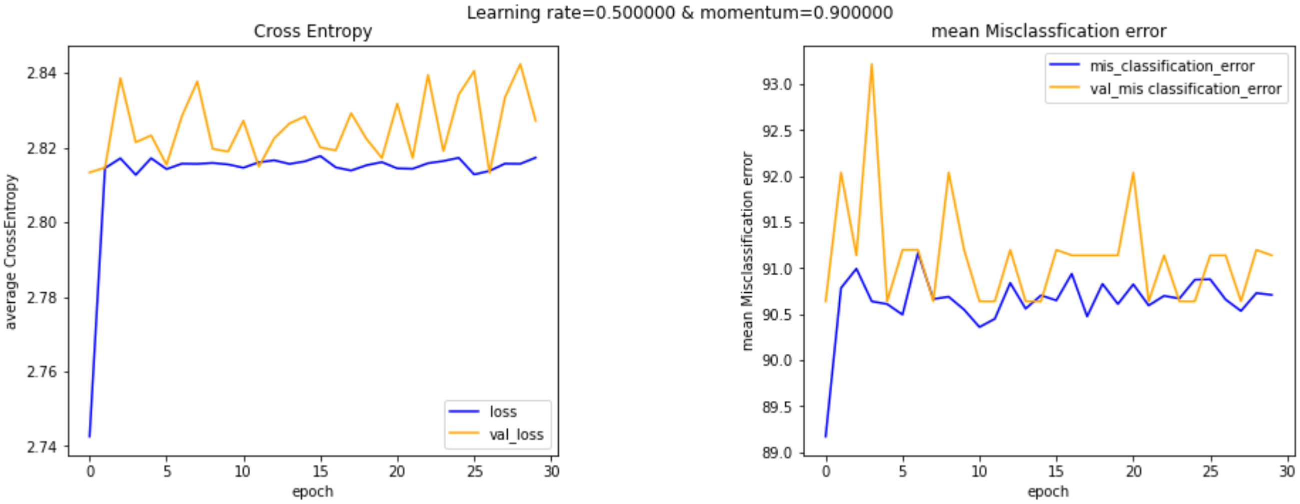
```
The misclassification error for validation set and test set are 90.640000% and
89.300000%.
```



```
The misclassification error for validation set and test set are 17.580003% and
12.840000%.
```



```
The misclassification error for validation set and test set are 22.460002% and
89.300000%.
```



```
The misclassification error for validation set and test set are 91.200000% and
90.980000%.
```

From above,we can see that some of mean misclassificaiton error are very unstable,and are fluctuated. When momentum = 0.9,the fluctuation is much more serious. And some of learning rate and momentum sets face overfitting problem. Compared with the the original data, the best value of the parameters is learning rate = 0.2, momentum = 0.5, with the misclassficaiton error 14.539999% and 11.360000%. The misclassification error for validation set and test set are both higher than the original data. The largest misclassficaiton error are 91.200000% and 90.980000%,where learning rate = 0.5, momentum = 0.9 For parameters visulization, most dark parts are distributed on sides. It is not too noisy, and too correlated.

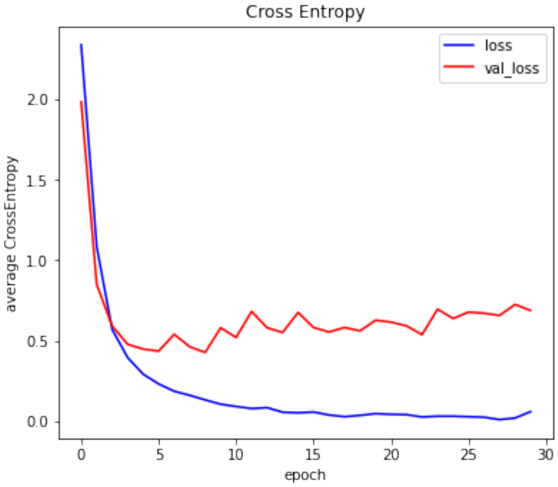
second:

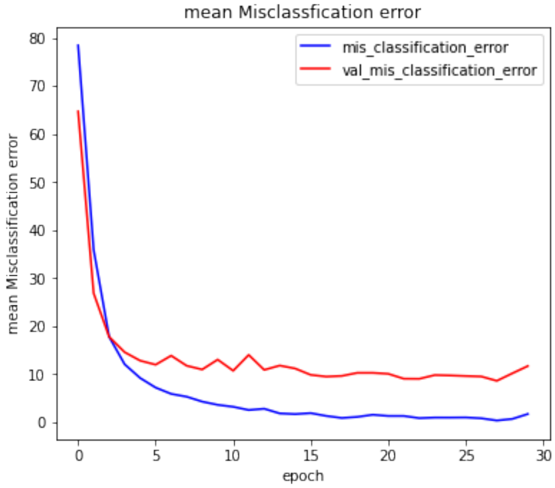
In [31]:

```
from tensorflow.keras.layers import BatchNormalization
```

In [33]:

```
np.random.seed(3)
sgd = keras.optimizers.SGD(learning_rate = 0.1, momentum = 0.5)
#Create single layer NN
model = Sequential()
#Adding Hidden layers
model.add(Conv2D(32, (3, 3), padding = 'same', activation='relu', input_shape
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dense(units = 19))
#Compiling the model
model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCrossentr
history = model.fit(X_train, Y_train, epochs = 30, validation_data = (X_val, 

#Plot
f, ax = plt.subplots(1, 2, figsize = (15, 5))
plt.subplots_adjust(left = 0.1, bottom = 0.1, right = 0.9, top = 0.9, wspace 
ax[0].plot(history.history['loss'], color = 'blue', label = "Training loss")
ax[0].plot(history.history['val_loss'], color = 'red', label = "validation lo
ax[0].set_xlabel('epoch')
ax[0].set_ylabel('average CrossEntropy')
ax[0].set_title('Cross Entropy')
ax[0].legend(['loss','val_loss'], fontsize = 10)

x = np.repeat(1, 30)
ax[1].plot((x - history.history['accuracy'])*100, color = 'blue', label = "mi
ax[1].plot((x - history.history['val_accuracy'])*100, color = 'red', label = 
ax[1].set_xlabel('epoch')
ax[1].set_ylabel('mean Misclassification error')
ax[1].set_title('mean Misclassfication error')
ax[1].legend(['mis_classification_error', 'val_mis_classification_error'], fo

plt.show()

pred = np.argmax(model.predict(X_test), axis=1)
err = np.mean(pred != Y_test)*100
print("The misclassification error for validation set and test set are %3f"%(
      'and %3f'%(err)+'%.')
```

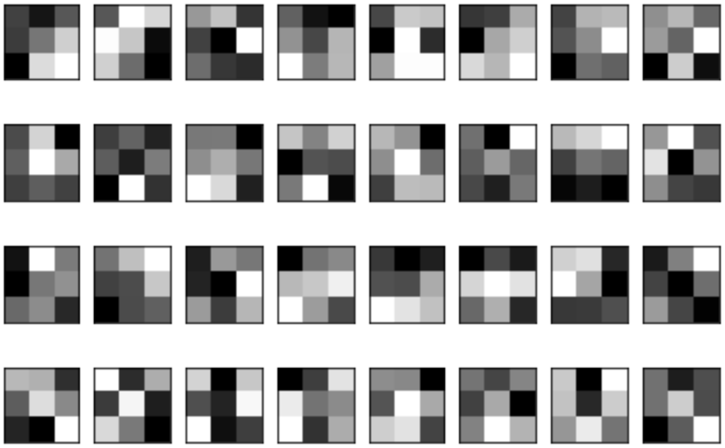




```
The misclassification error for validation set and test set are 13.000000% and
9.420000%.
```

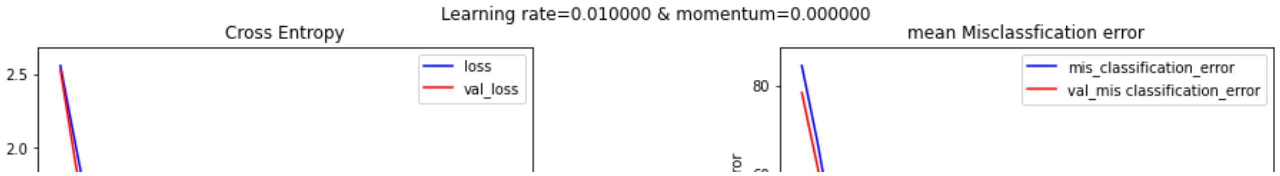
In [34]:

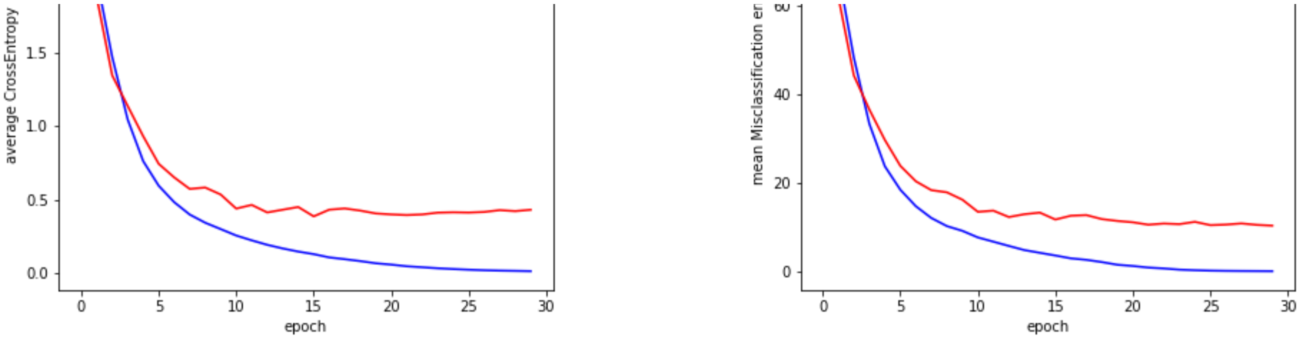
```
W = model.layers[0].get_weights()[0].reshape(3, 3, -1)
for i in range(W.shape[2]):
  plt.subplot(4, 8, i + 1)
  plt.imshow(W[:, :, i], cmap = 'gray')
  plt.xticks([])
  plt.yticks([])
plt.show()
```



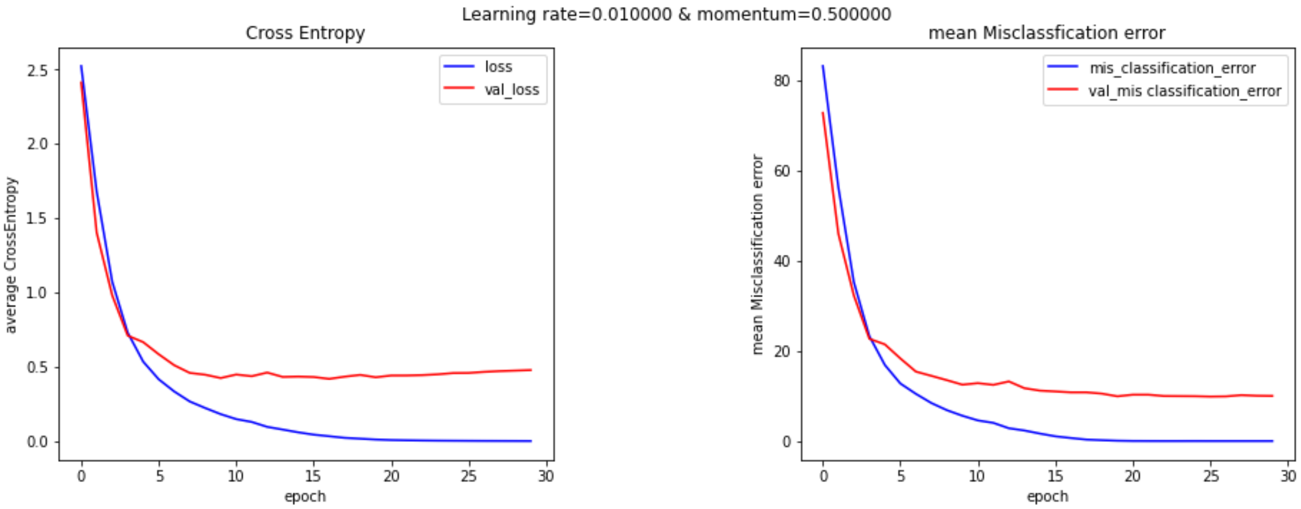
In [25]:

```
seed = 3

for lr in [0.01, 0.1, 0.2, 0.5]:
  for momentum in [0, 0.5, 0.9]:
    sgd = keras.optimizers.SGD(learning_rate = lr, momentum = momentum)
    np.random.seed(seed)

    #Create single layer NN
    model = Sequential()
    #Adding Hidden layer
    model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu', input_
    model.add(MaxPooling2D((2, 2)))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), activation="relu"))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(units = 64, activation = 'relu'))
    model.add(Dense(units = 19))
    #Compiling the model
    model.compile(optimizer = sgd, loss = keras.losses.SparseCategoricalCross
    history = model.fit(X_train, Y_train, epochs = 30,  validation_data = (X_

    #Plot
    f, ax = plt.subplots(1, 2, figsize = (15, 5))
    plt.subplots_adjust(left = 0.1, bottom = 0.1, right = 0.9, top = 0.9, wsp
    ax[0].plot(history.history['loss'], color = 'blue', label = "Training los
    ax[0].plot(history.history['val_loss'], color = 'red', label = "validatio
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('average CrossEntropy')
    ax[0].set_title('Cross Entropy')
    ax[0].legend(['loss', 'val_loss'], fontsize = 10)

    x = np.repeat(1, 30)
    ax[1].plot((x - history.history['accuracy'])*100, color = 'blue', label =
    ax[1].plot((x - history.history['val_accuracy'])*100, color = 'red', labe
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('mean Misclassification error')
    ax[1].set_title('mean Misclassfication error')
    ax[1].legend(['mis_classification_error', 'val_mis classification_error']

    f.suptitle('Learning rate=%2f & momentum=%2f'%(lr, momentum))
    plt.show()
    pred = np.argmax(model.predict(X_test), axis=1)
    err = np.mean(pred != Y_test)*100
    print("The misclassification error for validation set and test set are %3
          'and %3f'%(err)+'%.')
```

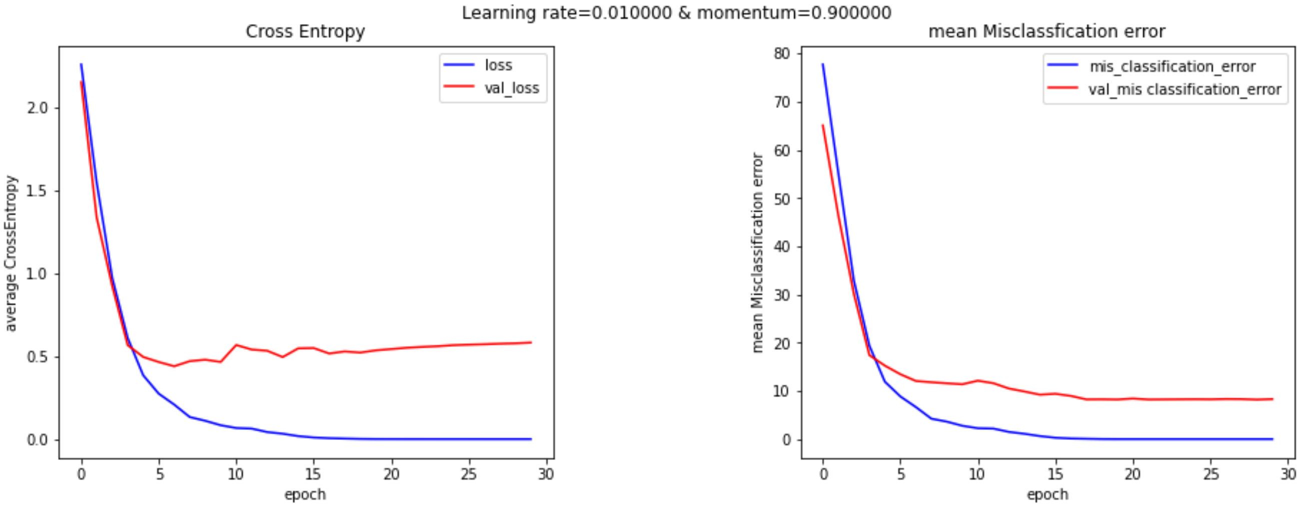




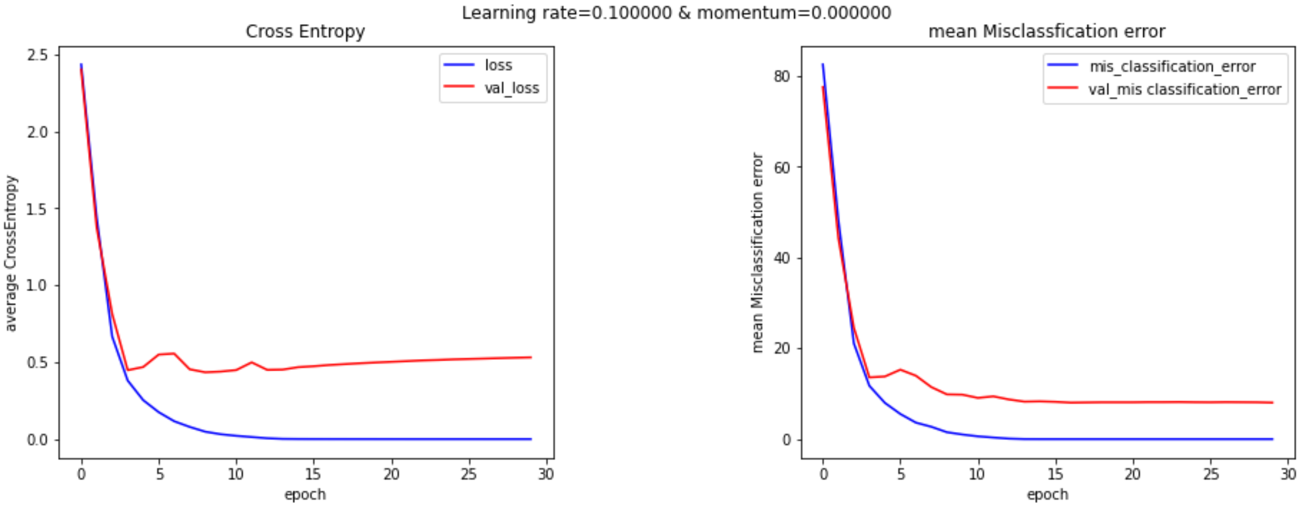
```
The misclassification error for validation set and test set are 16.140002% and
9.800000%.
```



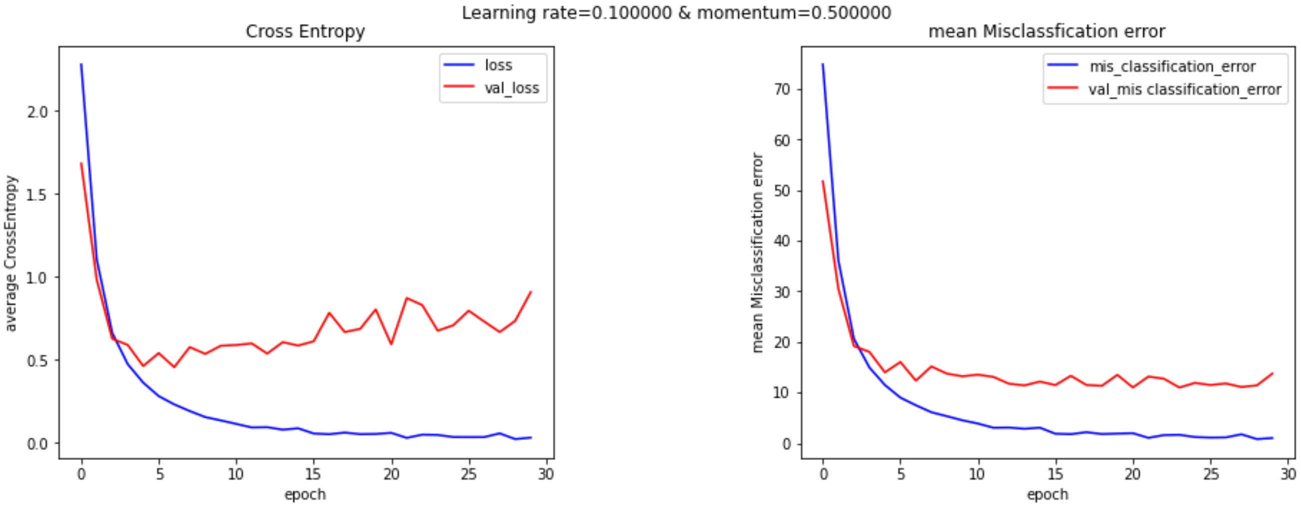
```
The misclassification error for validation set and test set are 12.500000% and
9.240000%.
```



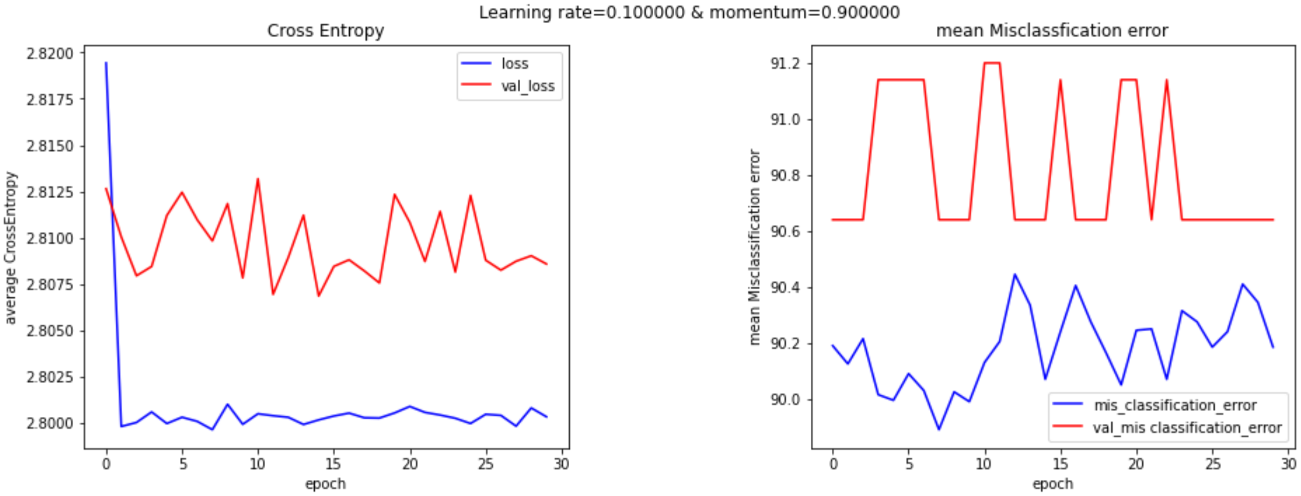
```
The misclassification error for validation set and test set are 11.379999% and
7.820000%.
```



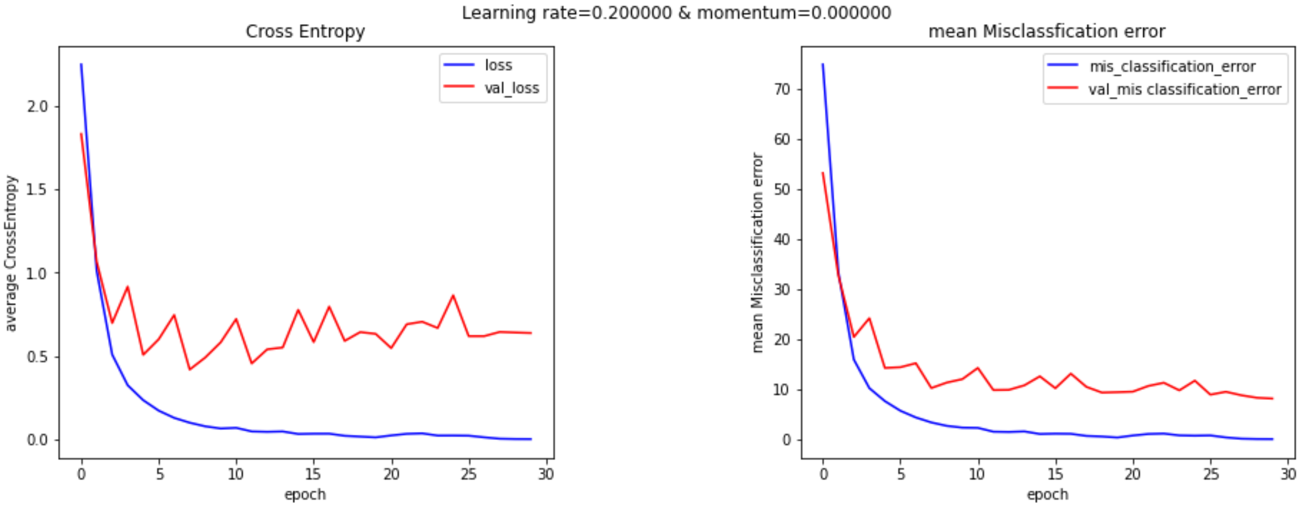
```
The misclassification error for validation set and test set are 9.799999% and
7.300000%.
```



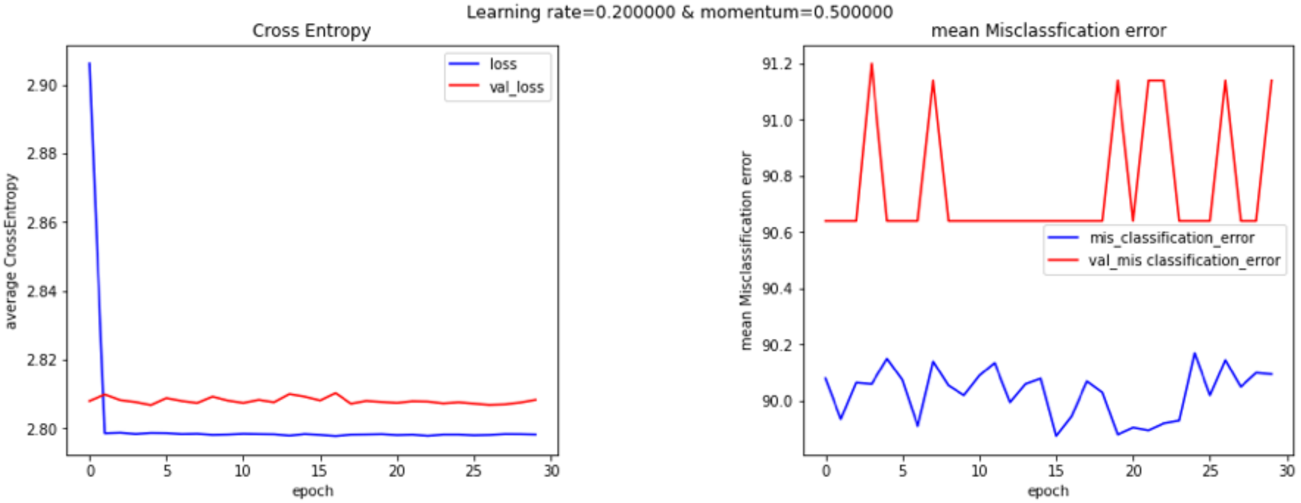
```
The misclassification error for validation set and test set are 13.220000% and
11.960000%.
```



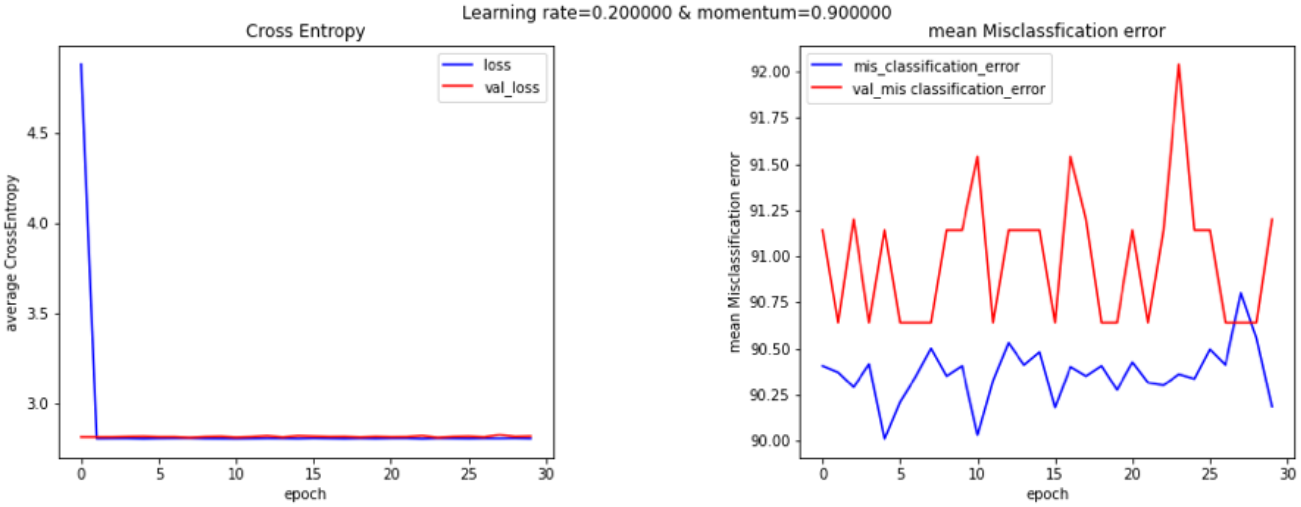
```
The misclassification error for validation set and test set are 90.640000% and
89.300000%.
```



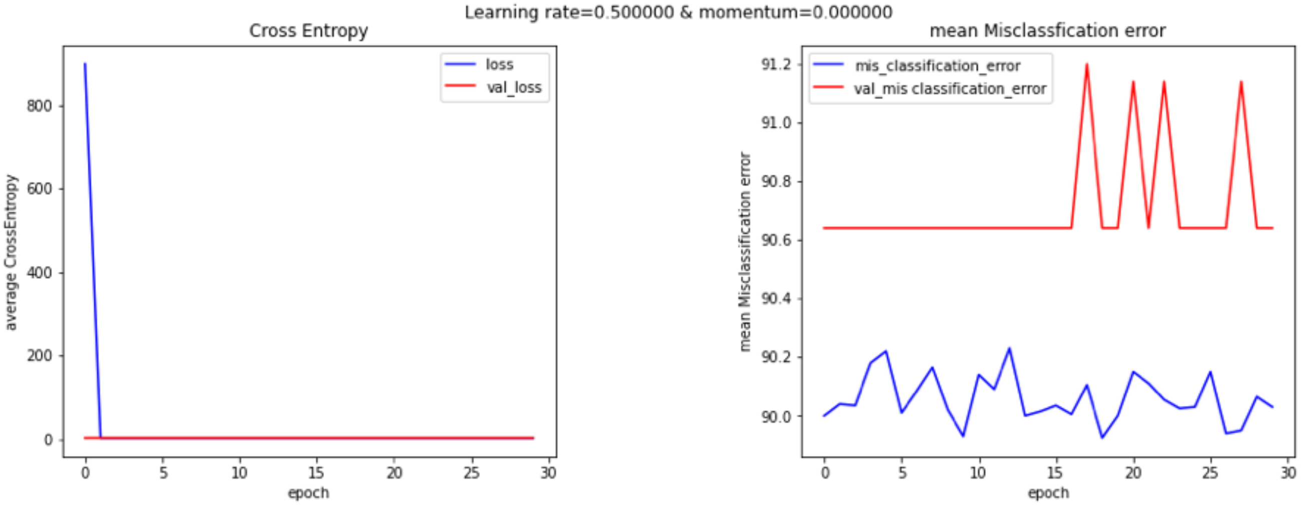
```
The misclassification error for validation set and test set are 11.979997% and
6.860000%.
```



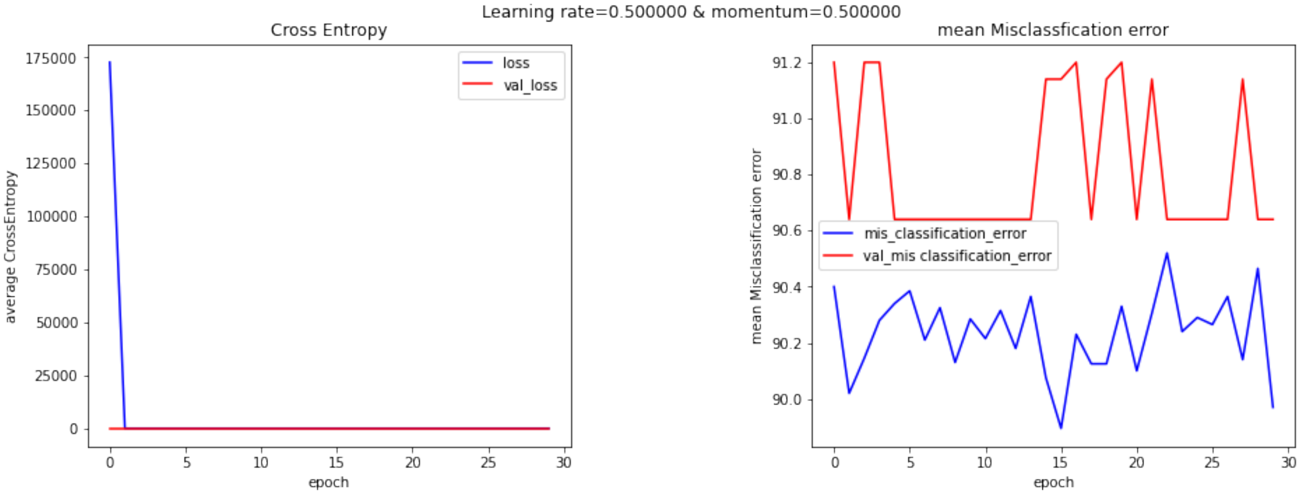
```
The misclassification error for validation set and test set are 90.640000% and
90.980000%.
```



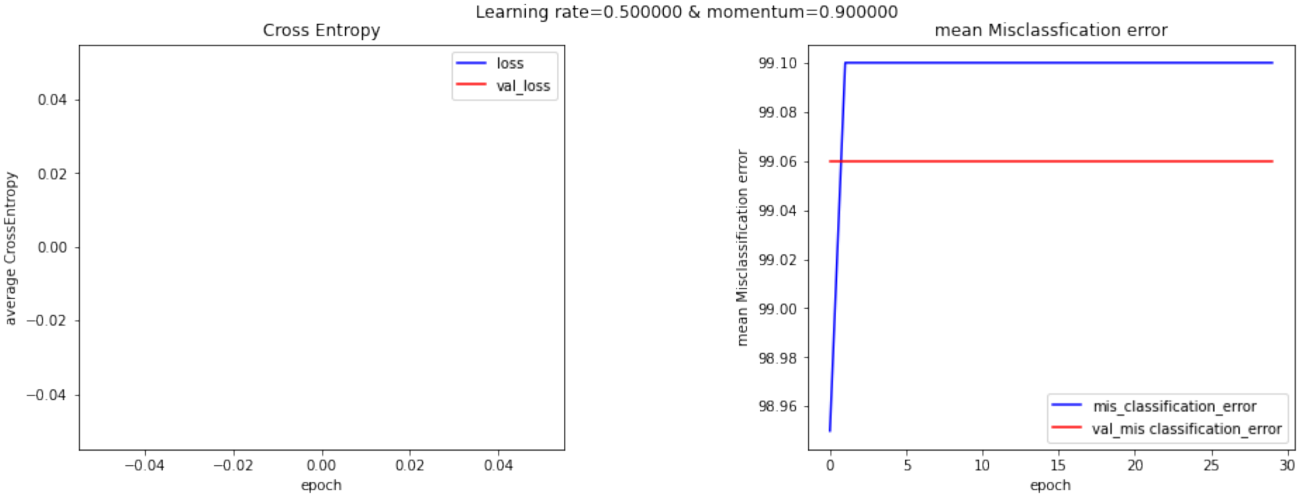
```
The misclassification error for validation set and test set are 91.140000% and
91.180000%.
```



```
The misclassification error for validation set and test set are 90.640000% and
89.300000%.
```



```
The misclassification error for validation set and test set are 90.640000% and
89.300000%.
```



```
The misclassification error for validation set and test set are 99.060000% and
99.120000%.
```

From above,we can see that most of mean misclassificaiton error are very unstable and when learning rate grows,the unstable situation is more clear. It fluctuated between different epoches and some misclassification error lines for vaalidation set and test set are event separate. Compared with the the original data, the best value of the parameters is learning rate = 0.1, momentum = 0.0, with the misclassficaiton error 9.799999% and 7.300000%. The misclassification error for validation set and test set are both much higher than the original data. For parameters visulization, most dark parts are distributed on sides. It is not too noisy, and too correlated.

why can't lower than 1%

Assuming that the neural network can perfectly calculate of the sum of the number and label of two images, not affected by the stitching of two images. The lowest misclassification error 0.669998%, with learning rate = 0.2 and momentum = 0.0. Thus, the probability of two correct classification is (1 - 0.669998%)(1 - 0.669998%) = 0.9866. Th, even if the assumption hold, it still cannot obtain a test error lower than 1%.

Reference: https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-inconvolutional-neural-networks/