Different architectures for CNN on MNIST dataset

1. Import Libraries

In [1]:

```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, BatchNormalization, Conv2D, MaxPooli
from keras import layers
from keras import backend as K
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
```

```
Using TensorFlow backend.
```

2. Global variables

In [39]:

```
batch_size = 128
num_classes = 10
epochs = 10
img_rows, img_cols, channels = 28, 28, 1
model_results = dict()
```

3. Load Data

In [3]:

```
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

3.1. Normalising Data

In [4]:

```
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (channels, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, channels)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

4.1. Model 1 - 2 Conv2D layers, 1 Max pool layer, 2 dense layers + dropout and flatten

In [5]:

```
model = Sequential()

#IN: (batch,rows,columns,channel)=>(batch,28,28,1)
#Remember the input_shape is specified without the batch
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
#OUT: (batch,rows,columns,channel)=>(batch,26,26,32)
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
#OUT: (batch,rows,columns,channel)=>(batch,24,24,64)
model.add(MaxPooling2D(pool_size=(2, 2)))
#OUT: (batch,rows,columns,channel)=>(batch,12,12,64)
model.add(Dropout(0.25))
model.add(Flatten())
#OUT: (batch,rows,columns,channel)=>(batch,9216)
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

```
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\tensorflow\python\framework\op_def_library.py:263: colocate_with (fro
m tensorflow.python.framework.ops) is deprecated and will be removed in a fu
ture version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\keras\backend\tensorflow_backend.py:3445: calling dropout (from tenso
rflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in
a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
```

In [6]:

```
1 model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
1 # model.get_config()
2 # model.inputs
3 # model.outputs
4 # model.get_weights()
```

In [8]:

```
1 model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.A
```

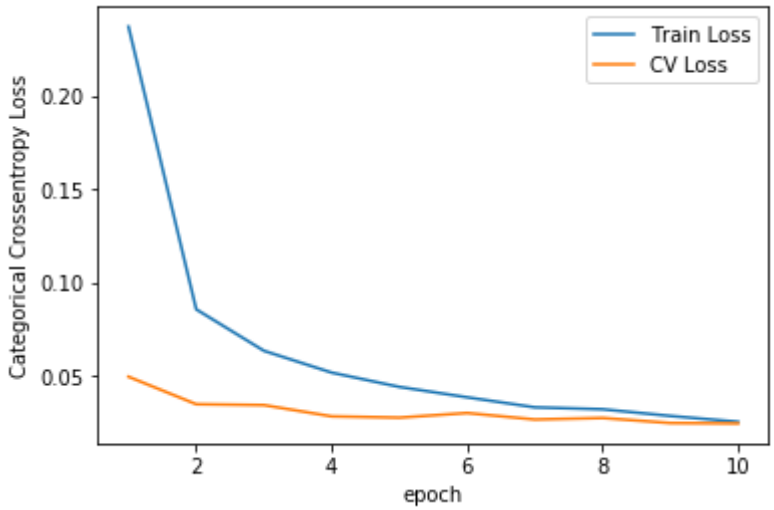
In [9]:

```
1 history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1,
```

```
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.pyt
hon.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 157s 3ms/step - loss: 0.2373
- acc: 0.9268 - val_loss: 0.0497 - val_acc: 0.9841
Epoch 2/10
60000/60000 [==============================] - 156s 3ms/step - loss: 0.0858
- acc: 0.9747 - val_loss: 0.0349 - val_acc: 0.9874
Epoch 3/10
60000/60000 [==============================] - 160s 3ms/step - loss: 0.0635
- acc: 0.9810 - val_loss: 0.0344 - val_acc: 0.9893
Epoch 4/10
60000/60000 [==============================] - 157s 3ms/step - loss: 0.0519
- acc: 0.9840 - val_loss: 0.0284 - val_acc: 0.9903
Epoch 5/10
60000/60000 [==============================] - 158s 3ms/step - loss: 0.0442
- acc: 0.9862 - val_loss: 0.0277 - val_acc: 0.9908
Epoch 6/10
60000/60000 [==============================] - 87s 1ms/step - loss: 0.0386 -
acc: 0.9878 - val_loss: 0.0302 - val_acc: 0.9902
Epoch 7/10
60000/60000 [==============================] - 84s 1ms/step - loss: 0.0332 -
acc: 0.9898 - val_loss: 0.0267 - val_acc: 0.9906
Epoch 8/10
60000/60000 [==============================] - 85s 1ms/step - loss: 0.0322 -
acc: 0.9901 - val_loss: 0.0275 - val_acc: 0.9909
Epoch 9/10
60000/60000 [==============================] - 86s 1ms/step - loss: 0.0286 -
acc: 0.9907 - val_loss: 0.0248 - val_acc: 0.9928
Epoch 10/10
60000/60000 [==============================] - 86s 1ms/step - loss: 0.0255 -
acc: 0.9920 - val_loss: 0.0246 - val_acc: 0.9931
```

In [10]:

```
x = list(range(1,epochs+1))
plt.plot(x, history.history['loss'],label='Train Loss')
plt.plot(x, history.history['val_loss'],label='CV Loss')
plt.xlabel('epoch')
plt.ylabel('Categorical Crossentropy Loss')
plt.legend()
plt.show()
```



In [11]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.02459182265629006
Test accuracy: 0.9931
```

In [12]:

```
score = model.evaluate(x_train, y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.004404588298801052
Train accuracy: 0.9988833333333333
```

In [40]:

```
model_results['model1'] = {'Test Loss': 0.0245,'Train Loss':0.0044}
```

4.2. Model 2 - 2 Conv2D layers, 2 Max pool layers, 2 dense layers

+ dropout,flatten and padding

In [14]:

```
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3,3), strides=(1,1), padding='same', activat
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(3,3), strides=(1,1), padding='same', activat
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())

model.add(Dense(units=100, activation=keras.activations.relu, kernel_initializer = ker

model.add(Dropout(0.5))

model.add(Dense(units=num_classes, activation=keras.activations.softmax, kernel_initia
```

In [15]:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_3 (Dense)              (None, 100)               156900
_________________________________________________________________
dropout_3 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1010
=================================================================
Total params: 167,478
Trainable params: 167,478
Non-trainable params: 0
_________________________________________________________________
```

In [16]:

```
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.A
```

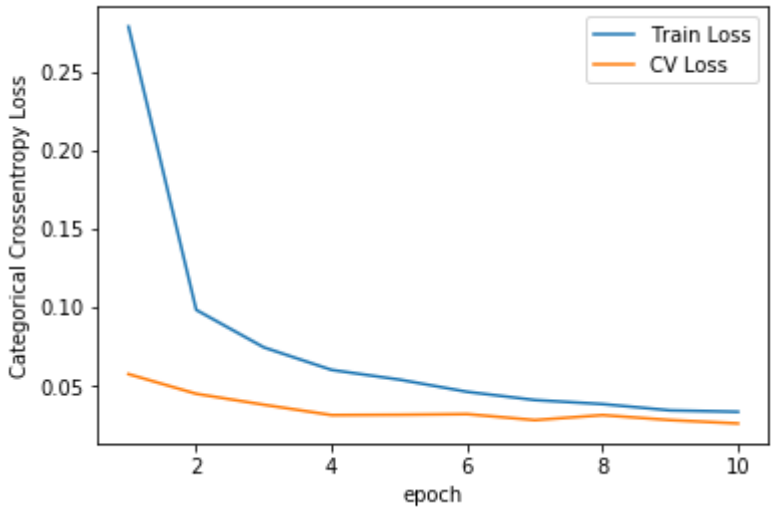
In [17]:

```
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1,
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 34s 567us/step - loss: 0.2788
- acc: 0.9140 - val_loss: 0.0575 - val_acc: 0.9817
Epoch 2/10
60000/60000 [==============================] - 33s 557us/step - loss: 0.0984
- acc: 0.9710 - val_loss: 0.0450 - val_acc: 0.9845
Epoch 3/10
60000/60000 [==============================] - 34s 569us/step - loss: 0.0745
- acc: 0.9779 - val_loss: 0.0380 - val_acc: 0.9877
Epoch 4/10
60000/60000 [==============================] - 34s 559us/step - loss: 0.0602
- acc: 0.9822 - val_loss: 0.0314 - val_acc: 0.9887
Epoch 5/10
60000/60000 [==============================] - 34s 559us/step - loss: 0.0540
- acc: 0.9839 - val_loss: 0.0316 - val_acc: 0.9905
Epoch 6/10
60000/60000 [==============================] - 34s 559us/step - loss: 0.0462
- acc: 0.9860 - val_loss: 0.0320 - val_acc: 0.9896
Epoch 7/10
60000/60000 [==============================] - 34s 563us/step - loss: 0.0410
- acc: 0.9872 - val_loss: 0.0282 - val_acc: 0.9910
Epoch 8/10
60000/60000 [==============================] - 34s 564us/step - loss: 0.0384
- acc: 0.9877 - val_loss: 0.0314 - val_acc: 0.9905
Epoch 9/10
60000/60000 [==============================] - 34s 562us/step - loss: 0.0344
- acc: 0.9892 - val_loss: 0.0282 - val_acc: 0.9898
Epoch 10/10
60000/60000 [==============================] - 34s 572us/step - loss: 0.0335
- acc: 0.9895 - val_loss: 0.0261 - val_acc: 0.9910
```

In [18]:

```
x = list(range(1,epochs+1))
plt.plot(x, history.history['loss'],label='Train Loss')
plt.plot(x, history.history['val_loss'],label='CV Loss')
plt.xlabel('epoch')
plt.ylabel('Categorical Crossentropy Loss')
plt.legend()
plt.show()
```



In [19]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.026057142214136592
Test accuracy: 0.991
```

In [20]:

```
score = model.evaluate(x_train, y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.009017451262661794
Train accuracy: 0.9971333333333333
```

In [41]:

```
model_results['model2'] = {'Test Loss': 0.0260,'Train Loss':0.0090}
```

4.3. Model 3 - 4 Conv2D layers, 2 Max pool layers, 2 dense layers

+ dropout,flatten,padding and batchnormalising

In [22]:

```
model = Sequential()

#======================CONV BLOCK 1============================
model.add(Conv2D(filters=64, kernel_size=(2,2), strides=(1,1), padding='same', activat
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init
model.add(BatchNormalization())
model.add(Conv2D(filters=64, kernel_size=(2,2), strides=(1,1), padding='valid', activa
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(3,3), strides=(3,3)))
#======================CONV BLOCK 1============================


#======================CONV BLOCK 2============================
model.add(Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), padding='same', activa
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), padding='same', activa
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(3,3), strides=(3,3)))
#======================CONV BLOCK 2============================


#======================DENSE============================
model.add(Flatten())
model.add(Dense(units=200, activation=keras.activations.relu, kernel_initializer = ker
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(units=num_classes, activation=keras.activations.softmax, kernel_initia
#======================DENSE============================
```

In [23]:

```
1 model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 28, 28, 64)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 28, 28, 64)        256
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 27, 27, 64)        16448
_________________________________________________________________
batch_normalization_2 (Batch (None, 27, 27, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 9, 9, 64)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 9, 9, 128)         73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 9, 9, 128)         512
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 9, 9, 128)         147584
_________________________________________________________________
batch_normalization_4 (Batch (None, 9, 9, 128)         512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_5 (Dense)              (None, 200)               230600
_________________________________________________________________
dropout_4 (Dropout)          (None, 200)               0
_________________________________________________________________
batch_normalization_5 (Batch (None, 200)               800
_________________________________________________________________
dense_6 (Dense)              (None, 10)                2010
=================================================================
Total params: 473,154
Trainable params: 471,986
Non-trainable params: 1,168
_________________________________________________________________
```

In [24]:

```
1 model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.A
```

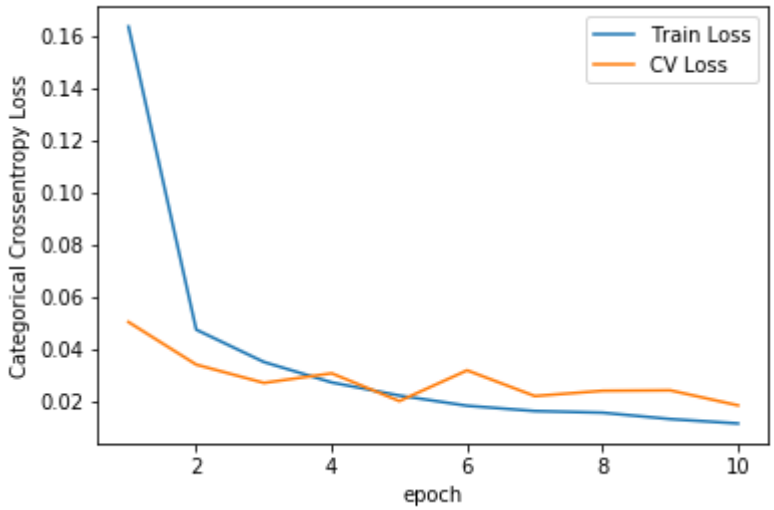
In [25]:

```
1 history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1,
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 554s 9ms/step - loss: 0.1633
- acc: 0.9551 - val_loss: 0.0503 - val_acc: 0.9844
Epoch 2/10
60000/60000 [==============================] - 552s 9ms/step - loss: 0.0473
- acc: 0.9877 - val_loss: 0.0340 - val_acc: 0.9884
Epoch 3/10
60000/60000 [==============================] - 551s 9ms/step - loss: 0.0350
- acc: 0.9904 - val_loss: 0.0270 - val_acc: 0.9907
Epoch 4/10
60000/60000 [==============================] - 553s 9ms/step - loss: 0.0271
- acc: 0.9920 - val_loss: 0.0306 - val_acc: 0.9906
Epoch 5/10
60000/60000 [==============================] - 552s 9ms/step - loss: 0.0221
- acc: 0.9939 - val_loss: 0.0200 - val_acc: 0.9935
Epoch 6/10
60000/60000 [==============================] - 552s 9ms/step - loss: 0.0182
- acc: 0.9947 - val_loss: 0.0318 - val_acc: 0.9898
Epoch 7/10
60000/60000 [==============================] - 552s 9ms/step - loss: 0.0162
- acc: 0.9950 - val_loss: 0.0219 - val_acc: 0.9934
Epoch 8/10
60000/60000 [==============================] - 556s 9ms/step - loss: 0.0156
- acc: 0.9952 - val_loss: 0.0239 - val_acc: 0.9921
Epoch 9/10
60000/60000 [==============================] - 552s 9ms/step - loss: 0.0131
- acc: 0.9960 - val_loss: 0.0241 - val_acc: 0.9917
Epoch 10/10
60000/60000 [==============================] - 553s 9ms/step - loss: 0.0114
- acc: 0.9966 - val_loss: 0.0184 - val_acc: 0.9944
```

In [26]:

```
x = list(range(1,epochs+1))
plt.plot(x, history.history['loss'],label='Train Loss')
plt.plot(x, history.history['val_loss'],label='CV Loss')
plt.xlabel('epoch')
plt.ylabel('Categorical Crossentropy Loss')
plt.legend()
plt.show()
```



In [27]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.01836347421152368
Test accuracy: 0.9944
```

In [28]:

```
score = model.evaluate(x_train, y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.003370044193206043
Train accuracy: 0.9988666666666667
```

In [42]:

```
model_results['model3'] = {'Test Loss': 0.0183,'Train Loss':0.0033}
```

I think the above model overfits somewhat due to the deep network

4.4. Model 4 - 2 Conv2D layers, 1 Max pool layers, 2 dense layers + dropout,flatten,padding,batchnormalising and stride

In [30]:

```
model = Sequential()

#======================CONV BLOCK 1=============================
model.add(Conv2D(filters=64, kernel_size=(2,2), strides=(2,2), padding='same', activat
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init
model.add(BatchNormalization())
model.add(Conv2D(filters=64, kernel_size=(4,4), strides=(2,2), padding='same', activat
                 kernel_initializer = keras.initializers.he_uniform(seed=1), bias_init
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
#======================CONV BLOCK 1=============================


#======================DENSE=============================
model.add(Flatten())
model.add(Dense(units=200, activation=keras.activations.relu, kernel_initializer = ker
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(units=num_classes, activation=keras.activations.softmax, kernel_initia
#======================DENSE=============================
```

In [31]:

```
1 model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 14, 14, 64)        320
_________________________________________________________________
batch_normalization_6 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 7, 7, 64)          65600
_________________________________________________________________
batch_normalization_7 (Batch (None, 7, 7, 64)          256
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_7 (Dense)              (None, 200)               115400
_________________________________________________________________
dropout_5 (Dropout)          (None, 200)               0
_________________________________________________________________
batch_normalization_8 (Batch (None, 200)               800
_________________________________________________________________
dense_8 (Dense)              (None, 10)                2010
=================================================================
Total params: 184,642
Trainable params: 183,986
Non-trainable params: 656
_________________________________________________________________
```

In [32]:

```
1 model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.A
```

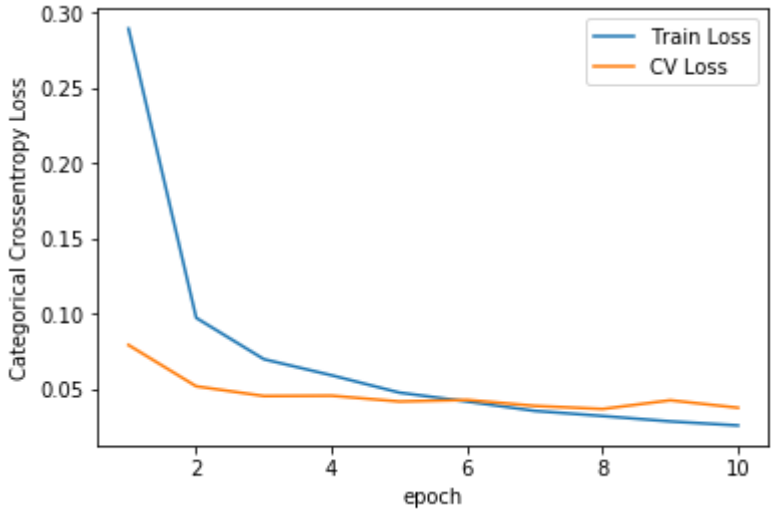
In [33]:

```
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1,
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.2895 -
acc: 0.9128 - val_loss: 0.0791 - val_acc: 0.9749
Epoch 2/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0971 -
acc: 0.9712 - val_loss: 0.0516 - val_acc: 0.9840
Epoch 3/10
60000/60000 [==============================] - 72s 1ms/step - loss: 0.0696 -
acc: 0.9787 - val_loss: 0.0451 - val_acc: 0.9852
Epoch 4/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0589 -
acc: 0.9821 - val_loss: 0.0454 - val_acc: 0.9838
Epoch 5/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0474 -
acc: 0.9857 - val_loss: 0.0415 - val_acc: 0.9865
Epoch 6/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0415 -
acc: 0.9871 - val_loss: 0.0426 - val_acc: 0.9860
Epoch 7/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0353 -
acc: 0.9886 - val_loss: 0.0386 - val_acc: 0.9880
Epoch 8/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0319 -
acc: 0.9901 - val_loss: 0.0365 - val_acc: 0.9887
Epoch 9/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0283 -
acc: 0.9911 - val_loss: 0.0423 - val_acc: 0.9880
Epoch 10/10
60000/60000 [==============================] - 73s 1ms/step - loss: 0.0256 -
acc: 0.9914 - val_loss: 0.0373 - val_acc: 0.9890
```

In [34]:

```
x = list(range(1,epochs+1))
plt.plot(x, history.history['loss'],label='Train Loss')
plt.plot(x, history.history['val_loss'],label='CV Loss')
plt.xlabel('epoch')
plt.ylabel('Categorical Crossentropy Loss')
plt.legend()
plt.show()
```



In [35]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.03733194876129637
Test accuracy: 0.989
```

In [36]:

```
score = model.evaluate(x_train, y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
```

```
Train loss: 0.007702990999433435
Train accuracy: 0.99775
```

In [45]:

```
model_results['model4'] = {'Test Loss': 0.0373,'Train Loss':0.0077}
```

5. Model Results

In [46]:

```
1 pd.DataFrame(model_results)
```

Out[46]:

	model1	model2	model3	model4
Test Loss	0.0245	0.026	0.0183	0.0373
Train Loss	0.0044	0.009	0.0033	0.0077