

In [1]:

```
1 from keras.layers import Conv2D, Dense, Dropout, MaxPooling2D, AveragePooling2D,
2 from keras.models import Sequential
3 from keras.datasets import mnist
4 from keras.utils import to_categorical
5 from keras.losses import categorical_crossentropy
6 import numpy as np
```

Using TensorFlow backend.

In [2]:

```
1 (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Downloading data from <https://s3.amazonaws.com/img-datasets/mnist.npz>
(<https://s3.amazonaws.com/img-datasets/mnist.npz>)
11493376/11490434 [=====] - 2s 0us/step

In [3]:

```
1 print(f'length of the train dataset is {x_train.shape[0]} and test dataset is {x_test.shape[0]}')
```

length of the train dataset is 60000 and test dataset is 10000

In [4]:

```
1 max_value = np.max(x_train[0])
2 x_train = x_train/max_value
3 x_test = x_test/max_value
```

In [5]:

```
1 ytr = to_categorical(y_train)
2 yte = to_categorical(y_test)
```

In [6]:

```
1 num_classes = len(ytr[0])
```

In [7]:

```
1 num_classes
```

Out[7]:

10

In [8]:

```
1 print(y_train[0])
2 ytr[0]
```

5

Out[8]:

array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)

In [9]:

```
1 x_train[0].shape
```

Out[9]:

(28, 28)

In [10]:

```
1 batch_size=100
2 epochs=10
```

In [11]:

```
1 from keras import backend as K
2 K.image_data_format()
```

Out[11]:

'channels_last'

In [12]:

```
1 img_rows, img_cols = 28 , 28
2 if K.image_data_format() == 'channels_first':
3     x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
4     x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
5     input_shape = (1, img_rows, img_cols)
6 else:
7     x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
8     x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
9     input_shape = (img_rows, img_cols, 1)
```

In [13]:

```
1 x_train.shape
```

Out[13]:

(60000, 28, 28, 1)

In [14]:

```
1 input_shape
```

Out[14]:

(28, 28, 1)

Convolution layers...

3 Conv Layers with kernel size 3

In [15]:

```
1 conv_1 = Conv2D(filters=32, kernel_size=(3,3), input_shape = (input_shape), acti
2 conv_2 = Conv2D(filters=64, kernel_size=(3,3), activation='relu')
3 conv_3 = Conv2D(filters=128, kernel_size=(3,3), activation='relu')
4
5 max_pool_1 = MaxPooling2D()
6 max_pool_2 = MaxPooling2D()
7 max_pool_3 = MaxPooling2D()
8
9 dropout_1 = Dropout(0.25)
10 dropout_2 = Dropout(0.50)
11
12 flatten = Flatten()
13 dense_1 = Dense(units=128, activation='relu')
14 final = Dense(units=num_classes, activation='softmax')
15
16
17 model = Sequential()
18 model.add(conv_1)
19 model.add(max_pool_1)
20
21 model.add(conv_2)
22 model.add(max_pool_2)
23 model.add(conv_3)
24 model.add(max_pool_3)
25 model.add(flatten)
26 model.add(dense_1)
27 model.add(dropout_1)
28 model.add(final)
```

In [16]:

```
1 model.compile(optimizer='sgd', loss=categorical_crossentropy, metrics=['accuracy'])
```

In [23]:

```

1 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
2                     validation_data=(x_test, yte), verbose=1)

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [=====] - 3s 54us/step - loss: 0.1193 - accuracy: 0.9635 - val_loss: 0.0897 - val_accuracy: 0.9724

Epoch 2/10

60000/60000 [=====] - 3s 52us/step - loss: 0.1150 - accuracy: 0.9652 - val_loss: 0.0847 - val_accuracy: 0.9742

Epoch 3/10

60000/60000 [=====] - 3s 53us/step - loss: 0.1085 - accuracy: 0.9669 - val_loss: 0.0809 - val_accuracy: 0.9753

Epoch 4/10

60000/60000 [=====] - 3s 52us/step - loss: 0.1024 - accuracy: 0.9679 - val_loss: 0.0773 - val_accuracy: 0.9742

Epoch 5/10

60000/60000 [=====] - 3s 53us/step - loss: 0.0977 - accuracy: 0.9703 - val_loss: 0.0785 - val_accuracy: 0.9764

Epoch 6/10

60000/60000 [=====] - 3s 50us/step - loss: 0.0940 - accuracy: 0.9708 - val_loss: 0.0812 - val_accuracy: 0.9740

Epoch 7/10

60000/60000 [=====] - 3s 50us/step - loss: 0.0902 - accuracy: 0.9730 - val_loss: 0.0722 - val_accuracy: 0.9769

Epoch 8/10

60000/60000 [=====] - 3s 52us/step - loss: 0.0883 - accuracy: 0.9729 - val_loss: 0.0758 - val_accuracy: 0.9757

Epoch 9/10

60000/60000 [=====] - 3s 52us/step - loss: 0.0843 - accuracy: 0.9739 - val_loss: 0.0670 - val_accuracy: 0.9795

Epoch 10/10

60000/60000 [=====] - 3s 51us/step - loss: 0.0827 - accuracy: 0.9747 - val_loss: 0.0713 - val_accuracy: 0.9778

In [18]:

```

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

```

Test loss: 0.09905982046369463

Test accuracy: 0.967199981212616

In [30]:

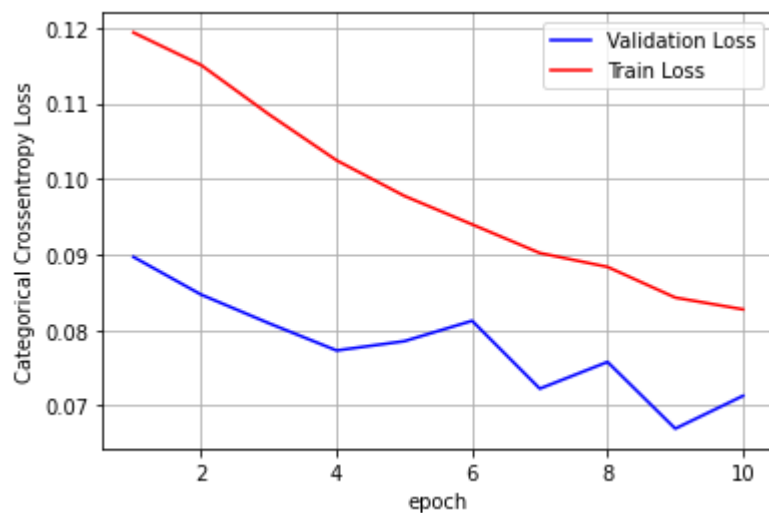
```

1 %matplotlib notebook
2 %matplotlib inline
3 import matplotlib.pyplot as plt
4 import time
5 def plt_dynamic(x, vy, ty, ax, colors=['b']):
6     ax.plot(x, vy, 'b', label="Validation Loss")
7     ax.plot(x, ty, 'r', label="Train Loss")
8     plt.legend()
9     plt.grid()
10    fig.canvas.draw()
11    # plt.show()

```

In [31]:

```
1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)
```



3 Conv Layers with Kernel size 5

In []:

```
1 Conv2D??
```

In [32]:

```

1 model = Sequential()
2 kernel=5
3 conv_1 = Conv2D(filters=32, kernel_size=kernel, input_shape = (input_shape), act
4 conv_2 = Conv2D(filters=64, kernel_size=kernel, activation='relu')
5 conv_3 = Conv2D(filters=128, kernel_size=kernel, activation='relu')
6
7 max_pool_1 = MaxPooling2D()
8 max_pool_2 = MaxPooling2D()
9 max_pool_3 = AveragePooling2D()
10
11 zero_pad = ZeroPadding2D(padding=(5,5))
12 dropout_1 = Dropout(0.25)
13
14 flatten = Flatten()
15 dense_1 = Dense(units=128, activation='relu')
16 final = Dense(units=num_classes, activation='softmax')
17
18
19 model.add(conv_1)
20 model.add(max_pool_1)
21 model.add(conv_2)
22 model.add(max_pool_2)
23 model.add(zero_pad)
24 model.add(conv_3)
25 model.add(flatten)
26 model.add(dense_1)
27 model.add(dropout_1)
28 model.add(final)
29
30 model.compile(optimizer='sgd', loss=categorical_crossentropy, metrics=['accuracy
31
32 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
33                     validation_data=(x_test, yte), verbose=1)
34
35 score = model.evaluate(x_test, yte, verbose=0)
36 print('Test loss:', score[0])
37 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [=====] - 5s 82us/step - loss: 1.
2277 - accuracy: 0.6570 - val_loss: 0.2906 - val_accuracy: 0.9157

Epoch 2/10

60000/60000 [=====] - 5s 78us/step - loss: 0.
2758 - accuracy: 0.9175 - val_loss: 0.1640 - val_accuracy: 0.9520

Epoch 3/10

60000/60000 [=====] - 5s 77us/step - loss: 0.
1801 - accuracy: 0.9461 - val_loss: 0.1142 - val_accuracy: 0.9664

Epoch 4/10

60000/60000 [=====] - 5s 77us/step - loss: 0.
1393 - accuracy: 0.9582 - val_loss: 0.0921 - val_accuracy: 0.9735

Epoch 5/10

60000/60000 [=====] - 5s 78us/step - loss: 0.
1164 - accuracy: 0.9647 - val_loss: 0.0752 - val_accuracy: 0.9764

Epoch 6/10

60000/60000 [=====] - 5s 77us/step - loss: 0.
1012 - accuracy: 0.9698 - val_loss: 0.0715 - val_accuracy: 0.9768

Epoch 7/10

60000/60000 [=====] - 5s 78us/step - loss: 0.

```

0896 - accuracy: 0.9730 - val_loss: 0.0635 - val_accuracy: 0.9801
Epoch 8/10
60000/60000 [=====] - 5s 78us/step - loss: 0.
0809 - accuracy: 0.9756 - val_loss: 0.0535 - val_accuracy: 0.9839
Epoch 9/10
60000/60000 [=====] - 5s 77us/step - loss: 0.
0738 - accuracy: 0.9779 - val_loss: 0.0511 - val_accuracy: 0.9831
Epoch 10/10
60000/60000 [=====] - 5s 77us/step - loss: 0.
0701 - accuracy: 0.9786 - val_loss: 0.0498 - val_accuracy: 0.9853
Test loss: 0.04984060045955702
Test accuracy: 0.9853000044822693

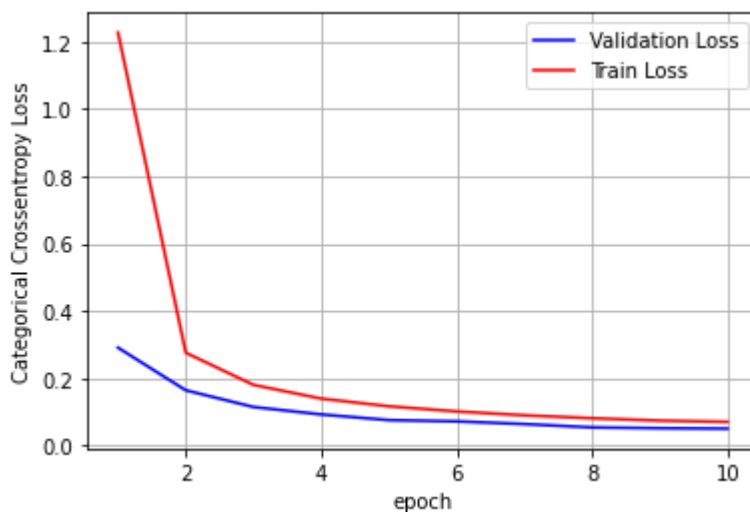
```

In [33]:

```

1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)

```



3 Conv Layers with Kernel Size 2

In [34]:

```

1 model = Sequential()
2 conv_1 = Conv2D(filters=32, kernel_size=(2,2), input_shape = (input_shape), acti
3 conv_2 = Conv2D(filters=63, kernel_size=(2,2), activation='relu')
4 conv_3 = Conv2D(filters=128, kernel_size=(2,2), activation='relu')
5
6 max_pool_1 = MaxPooling2D()
7 max_pool_2 = MaxPooling2D()
8
9 dropout_1 = Dropout(0.25)
10 dropout_2 = Dropout(0.50)
11
12 flatten = Flatten()
13 dense_1 = Dense(units=128, activation='relu')
14 final = Dense(units=num_classes, activation='softmax')
15
16
17 model.add(conv_1)
18 model.add(max_pool_1)
19 model.add(conv_2)
20 model.add(max_pool_2)
21 model.add(conv_3)
22 model.add(flatten)
23 model.add(dense_1)
24 model.add(dropout_1)
25 model.add(final)
26
27 model.compile(optimizer='sgd', loss=categorical_crossentropy, metrics=['accuracy
28
29 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
30                     validation_data=(x_test, yte), verbose=1)
31
32 score = model.evaluate(x_test, yte, verbose=0)
33 print('Test loss:', score[0])
34 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [=====] - 3s 52us/step - loss: 1.3356 - accuracy: 0.6007 - val_loss: 0.4047 - val_accuracy: 0.8833

Epoch 2/10

60000/60000 [=====] - 3s 50us/step - loss: 0.4045 - accuracy: 0.8752 - val_loss: 0.2539 - val_accuracy: 0.9231

Epoch 3/10

60000/60000 [=====] - 3s 48us/step - loss: 0.2977 - accuracy: 0.9074 - val_loss: 0.2042 - val_accuracy: 0.9359

Epoch 4/10

60000/60000 [=====] - 3s 49us/step - loss: 0.2373 - accuracy: 0.9272 - val_loss: 0.1671 - val_accuracy: 0.9498

Epoch 5/10

60000/60000 [=====] - 3s 50us/step - loss: 0.1972 - accuracy: 0.9390 - val_loss: 0.1380 - val_accuracy: 0.9596

Epoch 6/10

60000/60000 [=====] - 3s 52us/step - loss: 0.1707 - accuracy: 0.9484 - val_loss: 0.1162 - val_accuracy: 0.9660

Epoch 7/10

60000/60000 [=====] - 3s 50us/step - loss: 0.1517 - accuracy: 0.9533 - val_loss: 0.1048 - val_accuracy: 0.9662

Epoch 8/10


```

60000/60000 [=====] - 3s 51us/step - loss: 0.1378 - accuracy: 0.9578 - val_loss: 0.0965 - val_accuracy: 0.9706
Epoch 9/10
60000/60000 [=====] - 3s 49us/step - loss: 0.1235 - accuracy: 0.9626 - val_loss: 0.0931 - val_accuracy: 0.9721
Epoch 10/10
60000/60000 [=====] - 3s 50us/step - loss: 0.1151 - accuracy: 0.9649 - val_loss: 0.0826 - val_accuracy: 0.9741
Test loss: 0.08255197058618069
Test accuracy: 0.9740999937057495

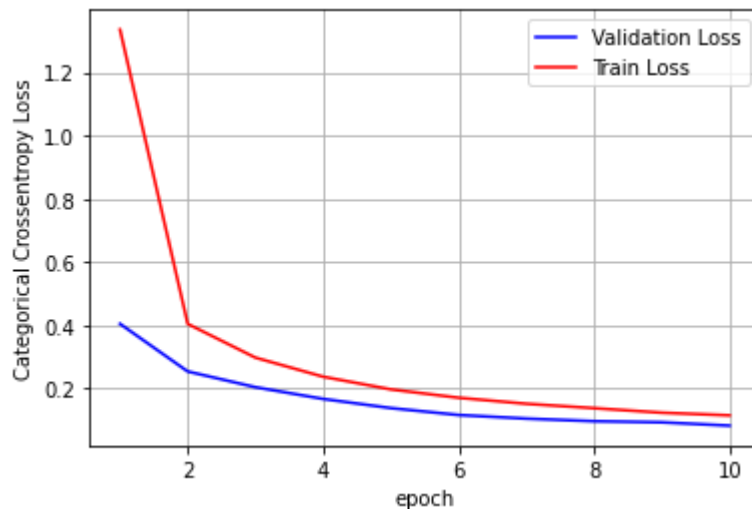
```

In [35]:

```

1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)

```



5 Convolutional Layers

Kernel Size=2

In [36]:

```

1 model = Sequential()
2 conv_1 = Conv2D(filters=32, kernel_size=(2,2), input_shape = input_shape, activation='relu')
3 conv_2 = Conv2D(filters=63, kernel_size=(2,2), activation='relu')
4 conv_3 = Conv2D(filters=128, kernel_size=(2,2), activation='relu')
5 conv_4 = Conv2D(filters=256, kernel_size=(2,2), activation='relu')
6 conv_5 = Conv2D(filters=512, kernel_size=(2,2), activation='relu')
7
8 zero_pad = ZeroPadding2D(padding=(5,5))
9
10 max_pool_1 = MaxPooling2D()
11 max_pool_2 = MaxPooling2D()
12
13 dropout_1 = Dropout(0.25)
14 dropout_2 = Dropout(0.50)
15
16 flatten = Flatten()
17 dense_1 = Dense(units=128, activation='relu')
18 final = Dense(units=num_classes, activation='softmax')
19
20 model.add(conv_1)
21 model.add(max_pool_1)
22 model.add(zero_pad)
23 model.add(conv_2)
24 model.add(max_pool_2)
25 model.add(conv_3)
26 model.add(conv_4)
27 model.add(conv_5)
28 model.add(flatten)
29 model.add(dense_1)
30 model.add(dropout_1)
31 model.add(final)
32
33 model.compile(optimizer='sgd', loss=categorical_crossentropy, metrics=['accuracy'])
34
35 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
36                     validation_data=(x_test, yte), verbose=1)
37
38 score = model.evaluate(x_test, yte, verbose=0)
39 print('Test loss:', score[0])
40 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [=====] - 10s 170us/step - loss: 1.4338 - accuracy: 0.5670 - val_loss: 0.3685 - val_accuracy: 0.8942

Epoch 2/10

60000/60000 [=====] - 10s 162us/step - loss: 0.3436 - accuracy: 0.8928 - val_loss: 0.2170 - val_accuracy: 0.9352

Epoch 3/10

60000/60000 [=====] - 10s 162us/step - loss: 0.2343 - accuracy: 0.9270 - val_loss: 0.1642 - val_accuracy: 0.9503

Epoch 4/10

60000/60000 [=====] - 10s 161us/step - loss: 0.1853 - accuracy: 0.9423 - val_loss: 0.1369 - val_accuracy: 0.9552

Epoch 5/10

60000/60000 [=====] - 10s 161us/step - loss: 0.1533 - accuracy: 0.9523 - val_loss: 0.1175 - val_accuracy: 0.9626

Epoch 6/10

60000/60000 [=====] - 10s 161us/step - loss:

```

0.1337 - accuracy: 0.9584 - val_loss: 0.1006 - val_accuracy: 0.9679
Epoch 7/10
60000/60000 [=====] - 10s 159us/step - loss:
0.1204 - accuracy: 0.9622 - val_loss: 0.0900 - val_accuracy: 0.9704
Epoch 8/10
60000/60000 [=====] - 10s 160us/step - loss:
0.1084 - accuracy: 0.9666 - val_loss: 0.0808 - val_accuracy: 0.9732
Epoch 9/10
60000/60000 [=====] - 10s 160us/step - loss:
0.0973 - accuracy: 0.9689 - val_loss: 0.0743 - val_accuracy: 0.9749
Epoch 10/10
60000/60000 [=====] - 10s 162us/step - loss:
0.0912 - accuracy: 0.9718 - val_loss: 0.0700 - val_accuracy: 0.9755
Test loss: 0.07003131841449067
Test accuracy: 0.9754999876022339

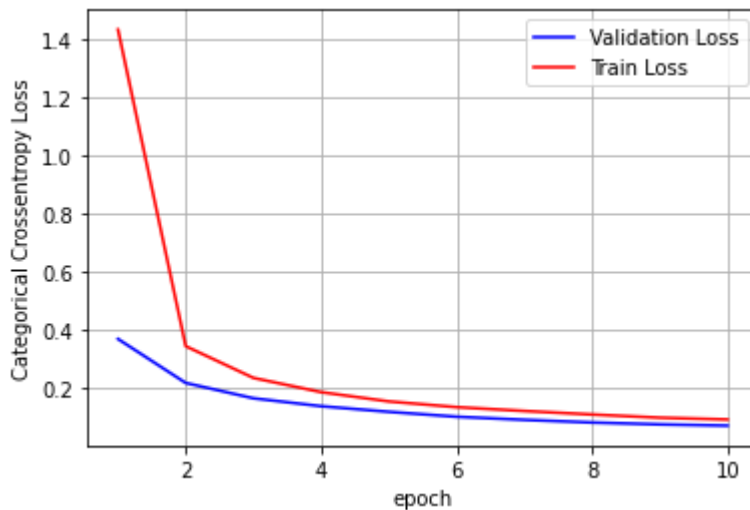
```

In [37]:

```

1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)

```



Kernel Size=3

In [40]:

```

1 kernel=3
2 model = Sequential()
3 conv_1 = Conv2D(filters=32, kernel_size=kernel, input_shape = input_shape, activation='relu')
4 conv_2 = Conv2D(filters=63, kernel_size=kernel, activation='relu')
5 conv_3 = Conv2D(filters=128, kernel_size=kernel, activation='relu')
6 conv_4 = Conv2D(filters=256, kernel_size=kernel, activation='relu')
7 conv_5 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
8
9 zero_pad = ZeroPadding2D(padding=(5,5))
10
11 max_pool_1 = MaxPooling2D()
12 max_pool_2 = MaxPooling2D()
13
14 dropout_1 = Dropout(0.25)
15 dropout_2 = Dropout(0.50)
16
17 flatten = Flatten()
18 dense_1 = Dense(units=128, activation='relu')
19 final = Dense(units=num_classes, activation='softmax')
20
21 model.add(conv_1)
22 model.add(max_pool_1)
23 model.add(zero_pad)
24 model.add(conv_2)
25 model.add(max_pool_2)
26 model.add(conv_3)
27 model.add(conv_4)
28 model.add(conv_5)
29 model.add(flatten)
30 model.add(dense_1)
31 model.add(dropout_1)
32 model.add(final)
33
34 model.compile(optimizer='sgd', loss=categorical_crossentropy, metrics=['accuracy'])
35
36 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
37                     validation_data=(x_test, yte), verbose=1)
38
39 score = model.evaluate(x_test, yte, verbose=0)
40 print('Test loss:', score[0])
41 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [=====] - 7s 122us/step - loss: 1.2687 - accuracy: 0.6658 - val_loss: 0.2930 - val_accuracy: 0.9082

Epoch 2/10

60000/60000 [=====] - 7s 121us/step - loss: 0.2785 - accuracy: 0.9132 - val_loss: 0.1680 - val_accuracy: 0.9490

Epoch 3/10

60000/60000 [=====] - 7s 121us/step - loss: 0.1837 - accuracy: 0.9434 - val_loss: 0.1193 - val_accuracy: 0.9618

Epoch 4/10

60000/60000 [=====] - 7s 123us/step - loss: 0.1396 - accuracy: 0.9568 - val_loss: 0.0923 - val_accuracy: 0.9690

Epoch 5/10

60000/60000 [=====] - 7s 122us/step - loss: 0.1143 - accuracy: 0.9649 - val_loss: 0.0767 - val_accuracy: 0.9754

Epoch 6/10

```

60000/60000 [=====] - 7s 121us/step - loss:
0.0960 - accuracy: 0.9703 - val_loss: 0.0696 - val_accuracy: 0.9770
Epoch 7/10
60000/60000 [=====] - 7s 119us/step - loss:
0.0823 - accuracy: 0.9740 - val_loss: 0.0648 - val_accuracy: 0.9788
Epoch 8/10
60000/60000 [=====] - 7s 120us/step - loss:
0.0749 - accuracy: 0.9764 - val_loss: 0.0537 - val_accuracy: 0.9830
Epoch 9/10
60000/60000 [=====] - 7s 120us/step - loss:
0.0664 - accuracy: 0.9793 - val_loss: 0.0505 - val_accuracy: 0.9817
Epoch 10/10
60000/60000 [=====] - 7s 120us/step - loss:
0.0601 - accuracy: 0.9810 - val_loss: 0.0475 - val_accuracy: 0.9845
Test loss: 0.047498973806994034
Test accuracy: 0.984499990940094

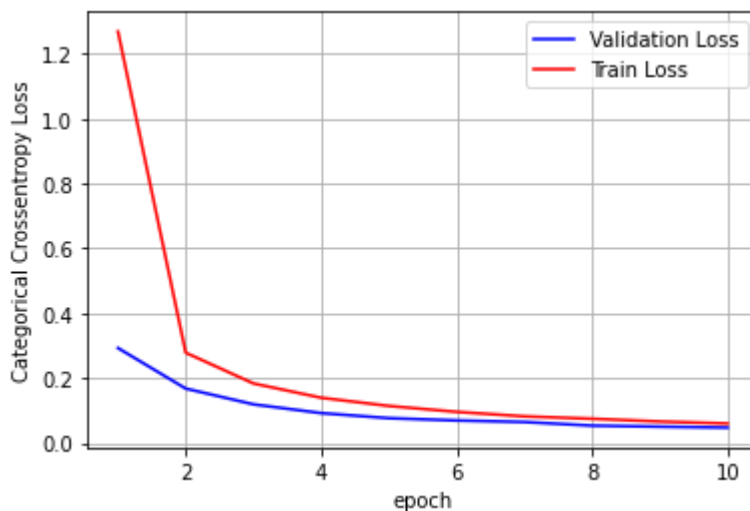
```

In [41]:

```

1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)

```



In [42]:

```

1 kernel=5
2 model = Sequential()
3 conv_1 = Conv2D(filters=32, kernel_size=kernel, input_shape = input_shape, activation='relu')
4 conv_2 = Conv2D(filters=63, kernel_size=kernel, activation='relu')
5 conv_3 = Conv2D(filters=128, kernel_size=kernel, activation='relu')
6 conv_4 = Conv2D(filters=256, kernel_size=kernel, activation='relu')
7 conv_5 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
8
9 zero_pad = ZeroPadding2D(padding=(10,10))
10
11 max_pool_1 = MaxPooling2D()
12 max_pool_2 = MaxPooling2D()
13
14 dropout_1 = Dropout(0.25)
15 dropout_2 = Dropout(0.50)
16
17 flatten = Flatten()
18 dense_1 = Dense(units=128, activation='relu')
19 final = Dense(units=num_classes, activation='softmax')
20
21 model.add(conv_1)
22 model.add(max_pool_1)
23 model.add(zero_pad)
24 model.add(conv_2)
25 model.add(max_pool_2)
26 model.add(conv_3)
27 model.add(conv_4)
28 model.add(conv_5)
29 model.add(flatten)
30 model.add(dense_1)
31 model.add(dropout_1)
32 model.add(final)
33
34 model.compile(optimizer='sgd', loss=categorical_crossentropy, metrics=['accuracy'])
35
36 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
37                     validation_data=(x_test, yte), verbose=1)
38
39 score = model.evaluate(x_test, yte, verbose=0)
40 print('Test loss:', score[0])
41 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [=====] - 16s 263us/step - loss: 0.9745 - accuracy: 0.7210 - val_loss: 0.2236 - val_accuracy: 0.9323

Epoch 2/10

60000/60000 [=====] - 15s 254us/step - loss: 0.2268 - accuracy: 0.9296 - val_loss: 0.1413 - val_accuracy: 0.9569

Epoch 3/10

60000/60000 [=====] - 15s 255us/step - loss: 0.1509 - accuracy: 0.9535 - val_loss: 0.0990 - val_accuracy: 0.9684

Epoch 4/10

60000/60000 [=====] - 15s 258us/step - loss: 0.1147 - accuracy: 0.9656 - val_loss: 0.0858 - val_accuracy: 0.9726

Epoch 5/10

60000/60000 [=====] - 16s 259us/step - loss: 0.0933 - accuracy: 0.9712 - val_loss: 0.0674 - val_accuracy: 0.9791

```

Epoch 6/10
60000/60000 [=====] - 15s 256us/step - loss:
0.0800 - accuracy: 0.9756 - val_loss: 0.0544 - val_accuracy: 0.9822
Epoch 7/10
60000/60000 [=====] - 15s 256us/step - loss:
0.0708 - accuracy: 0.9778 - val_loss: 0.0559 - val_accuracy: 0.9810
Epoch 8/10
60000/60000 [=====] - 16s 259us/step - loss:
0.0624 - accuracy: 0.9804 - val_loss: 0.0489 - val_accuracy: 0.9839
Epoch 9/10
60000/60000 [=====] - 15s 258us/step - loss:
0.0572 - accuracy: 0.9819 - val_loss: 0.0510 - val_accuracy: 0.9837
Epoch 10/10
60000/60000 [=====] - 15s 256us/step - loss:
0.0532 - accuracy: 0.9831 - val_loss: 0.0473 - val_accuracy: 0.9843
Test loss: 0.04731896648776019
Test accuracy: 0.9843000173568726

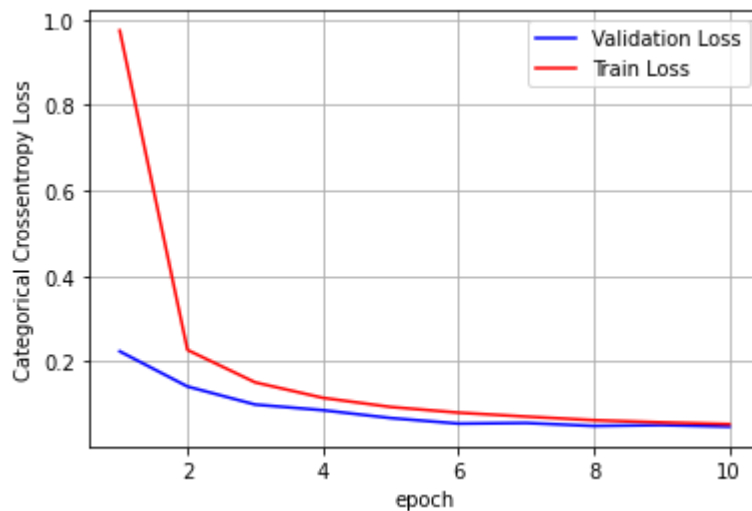
```

In [43]:

```

1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)

```



7 Convolutional Layers

Kernel size 2

In [44]:

```

1 kernel=2
2 epochs=4
3 model = Sequential()
4 conv_1 = Conv2D(filters=32, kernel_size=kernel, input_shape = input_shape, activation='relu')
5 conv_2 = Conv2D(filters=64, kernel_size=kernel, activation='relu')
6 conv_3 = Conv2D(filters=128, kernel_size=kernel, activation='relu')
7 conv_4 = Conv2D(filters=256, kernel_size=kernel, activation='relu')
8 conv_5 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
9 conv_6 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
10 conv_7 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
11
12 zero_pad = ZeroPadding2D(padding=(24,24))
13
14 max_pool_1 = MaxPooling2D()
15 max_pool_2 = MaxPooling2D()
16
17 dropout_1 = Dropout(0.25)
18 dropout_2 = Dropout(0.50)
19
20 flatten = Flatten()
21 dense_1 = Dense(units=128, activation='relu')
22 dense_2 = Dense(units=64, activation='relu')
23 final = Dense(units=num_classes, activation='softmax')
24
25 model.add(conv_1)
26 model.add(max_pool_1)
27 model.add(zero_pad)
28 model.add(conv_2)
29 model.add(max_pool_2)
30 model.add(conv_3)
31 model.add(conv_4)
32 model.add(conv_5)
33 model.add(conv_6)
34 model.add(conv_7)
35 model.add(flatten)
36 model.add(dense_1)
37 model.add(dropout_1)
38 model.add(dense_2)
39 model.add(dropout_2)
40 model.add(final)
41
42 model.compile(optimizer='adam', loss=categorical_crossentropy, metrics=['accuracy'])
43
44 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
45                     validation_data=(x_test, yte), verbose=1)
46
47 score = model.evaluate(x_test, yte, verbose=0)
48 print('Test loss:', score[0])
49 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/4

60000/60000 [=====] - 200s 3ms/step - loss: 0.4748 - accuracy: 0.8482 - val_loss: 0.0694 - val_accuracy: 0.9812

Epoch 2/4

60000/60000 [=====] - 196s 3ms/step - loss: 0.1302 - accuracy: 0.9695 - val_loss: 0.0571 - val_accuracy: 0.9841

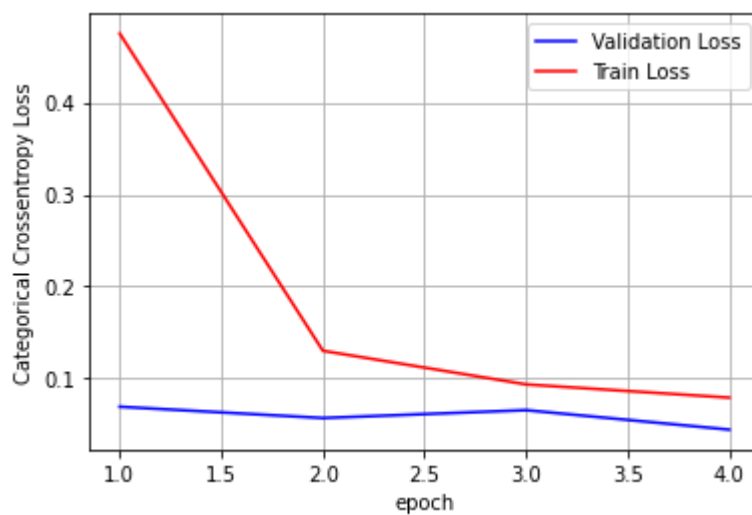
Epoch 3/4

60000/60000 [=====] - 196s 3ms/step - loss:

0.0936 - accuracy: 0.9779 - val_loss: 0.0657 - val_accuracy: 0.9831
Epoch 4/4
60000/60000 [=====] - 194s 3ms/step - loss:
0.0792 - accuracy: 0.9808 - val_loss: 0.0445 - val_accuracy: 0.9883
Test loss: 0.04449258501893973
Test accuracy: 0.9883000254631042

In [45]:

```
1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)
```



Kernel Size: 3

In [46]:

```
1 kernel=3
2 epochs=5
3 model = Sequential()
4 conv_1 = Conv2D(filters=32, kernel_size=kernel, input_shape = input_shape, activation='relu')
5 conv_2 = Conv2D(filters=64, kernel_size=kernel, activation='relu')
6 conv_3 = Conv2D(filters=128, kernel_size=kernel, activation='relu')
7 conv_4 = Conv2D(filters=256, kernel_size=kernel, activation='relu')
8 conv_5 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
9 conv_6 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
10 conv_7 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
11
12 zero_pad_1 = ZeroPadding2D(padding=(3,3))
13 zero_pad_2 = ZeroPadding2D(padding=(3,3))
14 zero_pad_3 = ZeroPadding2D(padding=(3,3))
15 zero_pad_4 = ZeroPadding2D(padding=(3,3))
16 zero_pad_5 = ZeroPadding2D(padding=(3,3))
17
18 max_pool_1 = MaxPooling2D()
19 max_pool_2 = MaxPooling2D()
20 max_pool_3 = MaxPooling2D()
21 max_pool_4 = MaxPooling2D()
22 max_pool_5 = MaxPooling2D()
23 max_pool_6 = MaxPooling2D()
24
25 dropout_1 = Dropout(0.25)
26 dropout_2 = Dropout(0.50)
27
28 flatten = Flatten()
29 dense_1 = Dense(units=128, activation='relu')
30 dense_2 = Dense(units=64, activation='relu')
31 final = Dense(units=num_classes, activation='softmax')
32
33 model.add(conv_1)
34 model.add(max_pool_1)
35 model.add(conv_2)
36 model.add(max_pool_2)
37 model.add(conv_3)
38 model.add(max_pool_3)
39 model.add(zero_pad_4)
40 model.add(conv_4)
41 model.add(max_pool_4)
42 model.add(zero_pad_5)
43 model.add(conv_5)
44 model.add(max_pool_5)
45 model.add(zero_pad_1)
46 model.add(conv_6)
47 model.add(max_pool_6)
48 model.add(zero_pad_2)
49 model.add(conv_7)
50 model.add(zero_pad_3)
51 model.add(flatten)
52 model.add(dense_1)
53 model.add(dropout_1)
54 model.add(dense_2)
55 model.add(dropout_2)
56 model.add(final)
57
58 model.compile(optimizer='adam', loss=categorical_crossentropy, metrics=['accuracy'])
59
```

```

60 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
61                     validation_data=(x_test, yte), verbose=1)
62
63 score = model.evaluate(x_test, yte, verbose=0)
64 print('Test loss:', score[0])
65 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/5

60000/60000 [=====] - 31s 517us/step - loss: 1.3812 - accuracy: 0.4629 - val_loss: 0.2774 - val_accuracy: 0.9295

Epoch 2/5

60000/60000 [=====] - 30s 499us/step - loss: 0.2413 - accuracy: 0.9414 - val_loss: 0.1078 - val_accuracy: 0.9721

Epoch 3/5

60000/60000 [=====] - 30s 498us/step - loss: 0.1457 - accuracy: 0.9674 - val_loss: 0.1172 - val_accuracy: 0.9754

Epoch 4/5

60000/60000 [=====] - 30s 498us/step - loss: 0.1144 - accuracy: 0.9747 - val_loss: 0.0893 - val_accuracy: 0.9799

Epoch 5/5

60000/60000 [=====] - 30s 498us/step - loss: 0.0922 - accuracy: 0.9792 - val_loss: 0.0950 - val_accuracy: 0.9823

Test loss: 0.09495201612031369

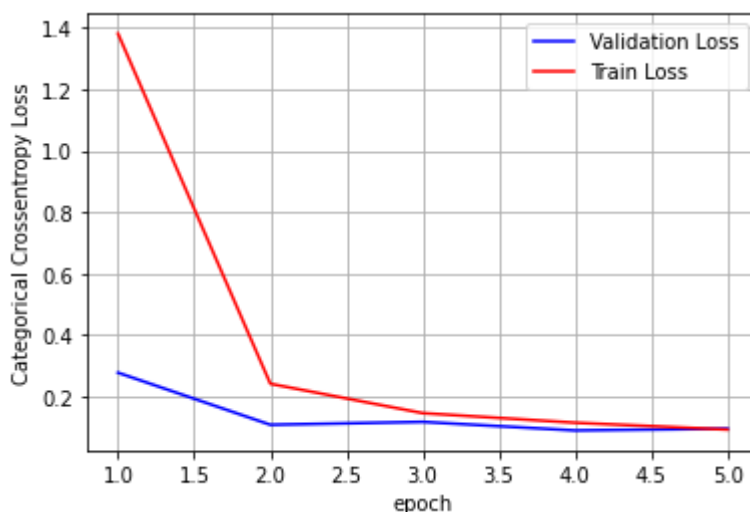
Test accuracy: 0.9822999835014343

In [47]:

```

1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)

```



Kernel Size: 5

In [48]:

```
1 kernel=5
2 epochs=5
3 model = Sequential()
4 conv_1 = Conv2D(filters=32, kernel_size=kernel, input_shape = input_shape, activation='relu')
5 conv_2 = Conv2D(filters=64, kernel_size=kernel, activation='relu')
6 conv_3 = Conv2D(filters=128, kernel_size=kernel, activation='relu')
7 conv_4 = Conv2D(filters=256, kernel_size=kernel, activation='relu')
8 conv_5 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
9 conv_6 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
10 conv_7 = Conv2D(filters=512, kernel_size=kernel, activation='relu')
11
12 zero_pad_0 = ZeroPadding2D(padding=(5,5))
13 zero_pad_1 = ZeroPadding2D(padding=(5,5))
14 zero_pad_2 = ZeroPadding2D(padding=(5,5))
15 zero_pad_3 = ZeroPadding2D(padding=(5,5))
16 zero_pad_4 = ZeroPadding2D(padding=(5,5))
17 zero_pad_5 = ZeroPadding2D(padding=(5,5))
18
19 max_pool_1 = MaxPooling2D()
20 avg_pool_2 = AveragePooling2D()
21 max_pool_3 = MaxPooling2D()
22 avg_pool_4 = AveragePooling2D()
23 max_pool_5 = MaxPooling2D()
24 max_pool_6 = MaxPooling2D()
25
26 dropout_1 = Dropout(0.25)
27 dropout_2 = Dropout(0.50)
28
29 flatten = Flatten()
30 dense_1 = Dense(units=128, activation='relu')
31 dense_2 = Dense(units=64, activation='relu')
32 dense_3 = Dense(units=32, activation='relu')
33 final = Dense(units=num_classes, activation='softmax')
34
35 model.add(conv_1)
36 model.add(max_pool_1)
37 model.add(conv_2)
38 model.add(avg_pool_2)
39 model.add(zero_pad_0)
40 model.add(conv_3)
41 model.add(max_pool_3)
42 model.add(zero_pad_1)
43 model.add(conv_4)
44 model.add(avg_pool_4)
45 model.add(zero_pad_2)
46 model.add(conv_5)
47 model.add(max_pool_5)
48 model.add(zero_pad_3)
49 model.add(conv_6)
50 model.add(max_pool_6)
51 model.add(zero_pad_4)
52 model.add(conv_7)
53 model.add(zero_pad_5)
54 model.add(flatten)
55 model.add(dense_1)
56 model.add(dense_2)
57 model.add(dense_3)
58 model.add(final)
59
```

```

60 model.compile(optimizer='sgd', loss=categorical_crossentropy, metrics=['accuracy'])
61
62 history = model.fit(x=x_train, y=ytr, batch_size=batch_size, epochs=epochs,
63                     validation_data=(x_test, yte), verbose=1)
64
65 score = model.evaluate(x_test, yte, verbose=0)
66 print('Test loss:', score[0])
67 print('Test accuracy:', score[1])

```

Train on 60000 samples, validate on 10000 samples

Epoch 1/5

60000/60000 [=====] - 96s 2ms/step - loss: 2.3019 - accuracy: 0.1118 - val_loss: 2.3015 - val_accuracy: 0.1135

Epoch 2/5

60000/60000 [=====] - 94s 2ms/step - loss: 2.3013 - accuracy: 0.1124 - val_loss: 2.3011 - val_accuracy: 0.1135

Epoch 3/5

60000/60000 [=====] - 94s 2ms/step - loss: 2.3011 - accuracy: 0.1124 - val_loss: 2.3009 - val_accuracy: 0.1135

Epoch 4/5

60000/60000 [=====] - 94s 2ms/step - loss: 2.3010 - accuracy: 0.1124 - val_loss: 2.3007 - val_accuracy: 0.1135

Epoch 5/5

60000/60000 [=====] - 94s 2ms/step - loss: 2.3008 - accuracy: 0.1124 - val_loss: 2.3006 - val_accuracy: 0.1135

Test loss: 2.3005532260894777

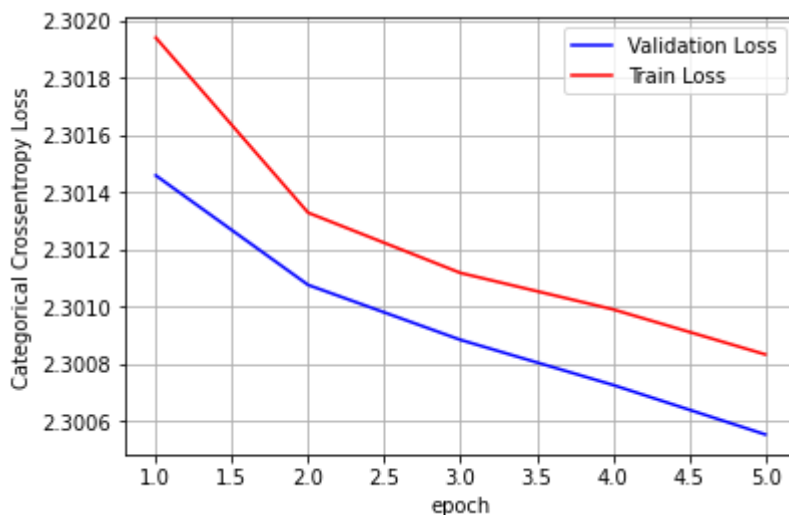
Test accuracy: 0.11349999904632568

In [49]:

```

1 fig,ax = plt.subplots(1,1)
2 ax.set_xlabel('epoch')
3 ax.set_ylabel('Categorical Crossentropy Loss')
4 # list of epoch numbers
5 x = list(range(1,epochs+1))
6 vy = history.history['val_loss']
7 ty = history.history['loss']
8 plt_dynamic(x, vy, ty, ax)

```



In []:

1

