Implementation of various CNN networks over MNIST dataset

Importing the necessary libraries

In [0]:
```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

import warnings
from sklearn.exceptions import DataConversionWarning
warnings.filterwarnings(action='ignore', category=DataConversionWarning)

# For plotting purposes
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from keras.utils import to_categorical
from keras.utils import np_utils
from keras.initializers import RandomNormal
```

Preparing the MNIST dataset

In [3]:
```
batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

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

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 = (img_cols, img_rows,1)
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, 1)

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)
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [0]:
```
# Plot train and cross validation loss
def plot_train_cv_loss(trained_model, epochs, colors=['b']):
    fig, ax = plt.subplots(1,1)
    ax.set_xlabel('epoch')
    ax.set_ylabel('Categorical Crossentropy Loss')
    x_axis_values = list(range(1,epochs+1))

    validation_loss = trained_model.history['val_loss']
    train_loss = trained_model.history['loss']

    ax.plot(x_axis_values, validation_loss, 'b', label="Validation Loss")
    ax.plot(x_axis_values, train_loss, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()


# Plot weight distribution using violin plot
def plot_weights(model):
    w_after = model.get_weights()

    o1_w = w_after[0].flatten().reshape(-1,1)
    o2_w = w_after[2].flatten().reshape(-1,1)
    out_w = w_after[4].flatten().reshape(-1,1)


    fig = plt.figure(figsize=(10,7))
    plt.title("Weight matrices after model trained\n")
    plt.subplot(1, 3, 1)
    plt.title("Trained model\n Weights")
    ax = sns.violinplot(y=o1_w,color='b')
    plt.xlabel('Hidden Layer 1')

    plt.subplot(1, 3, 2)
    plt.title("Trained model\n Weights")
    ax = sns.violinplot(y=o2_w, color='r')
    plt.xlabel('Hidden Layer 2 ')

    plt.subplot(1, 3, 3)
    plt.title("Trained model\n Weights")
    ax = sns.violinplot(y=out_w,color='y')
    plt.xlabel('Output Layer ')
    plt.show()
```

Basic single layer CNN network architecture

In [13]:
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.50))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.50))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
```

Training the CNN model

In [14]:
```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

single_CNN=model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

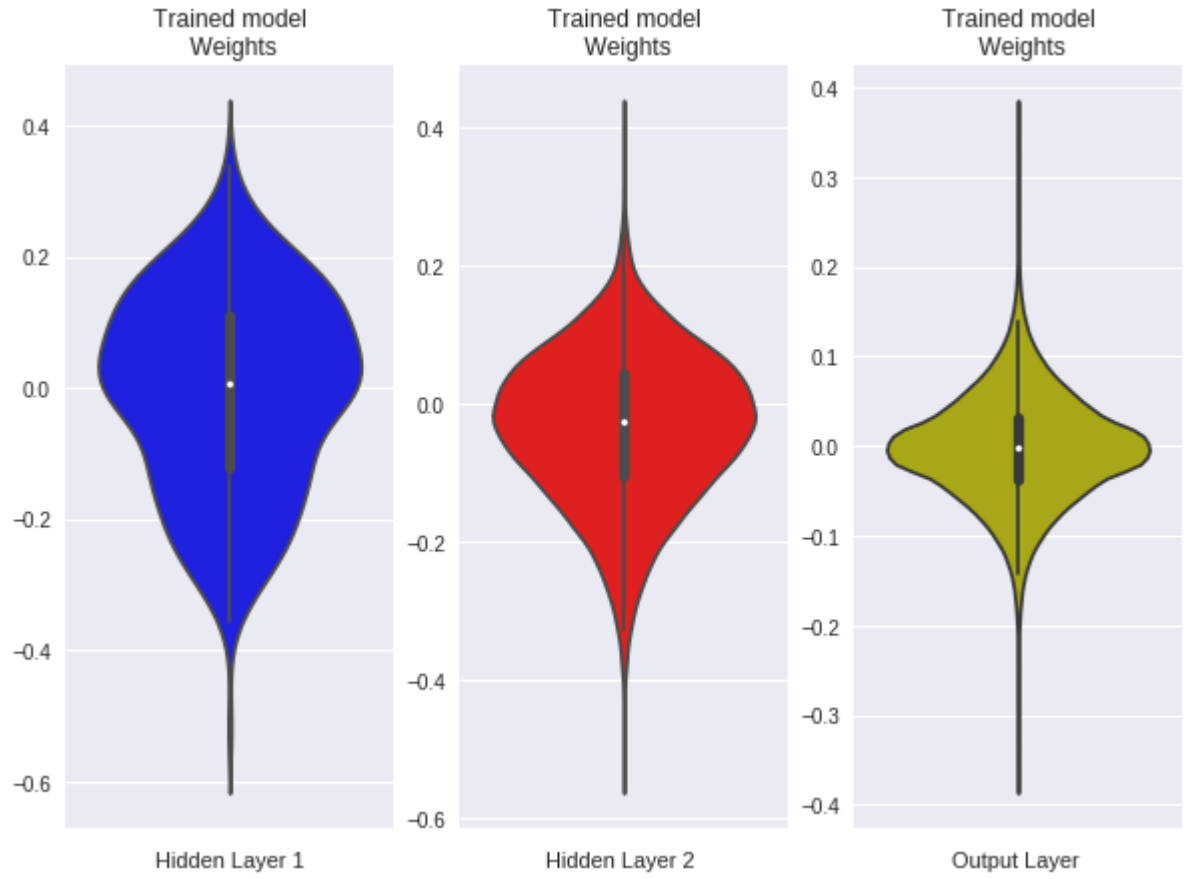
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 10s 159us/step - loss: 0.2537 - acc: 0.9216 - val_loss:
0.0530 - val_acc: 0.9814
Epoch 2/12
60000/60000 [==============================] - 9s 150us/step - loss: 0.0907 - acc: 0.9732 - val_loss:
0.0378 - val_acc: 0.9881
Epoch 3/12
60000/60000 [==============================] - 9s 150us/step - loss: 0.0703 - acc: 0.9785 - val_loss:
0.0329 - val_acc: 0.9891
Epoch 4/12
60000/60000 [==============================] - 9s 152us/step - loss: 0.0600 - acc: 0.9820 - val_loss:
0.0299 - val_acc: 0.9903
Epoch 5/12
60000/60000 [==============================] - 9s 152us/step - loss: 0.0504 - acc: 0.9844 - val_loss:
0.0310 - val_acc: 0.9904
Epoch 6/12
60000/60000 [==============================] - 9s 152us/step - loss: 0.0450 - acc: 0.9859 - val_loss:
0.0273 - val_acc: 0.9913
Epoch 7/12
60000/60000 [==============================] - 9s 152us/step - loss: 0.0417 - acc: 0.9868 - val_loss:
0.0257 - val_acc: 0.9917
Epoch 8/12
60000/60000 [==============================] - 9s 152us/step - loss: 0.0393 - acc: 0.9875 - val_loss:
0.0247 - val_acc: 0.9917
Epoch 9/12
60000/60000 [==============================] - 9s 152us/step - loss: 0.0357 - acc: 0.9887 - val_loss:
0.0263 - val_acc: 0.9913
Epoch 10/12
60000/60000 [==============================] - 9s 151us/step - loss: 0.0354 - acc: 0.9884 - val_loss:
0.0247 - val_acc: 0.9927
Epoch 11/12
60000/60000 [==============================] - 9s 151us/step - loss: 0.0310 - acc: 0.9903 - val_loss:
0.0275 - val_acc: 0.9918
Epoch 12/12
60000/60000 [==============================] - 9s 152us/step - loss: 0.0324 - acc: 0.9895 - val_loss:
0.0264 - val_acc: 0.9916
```

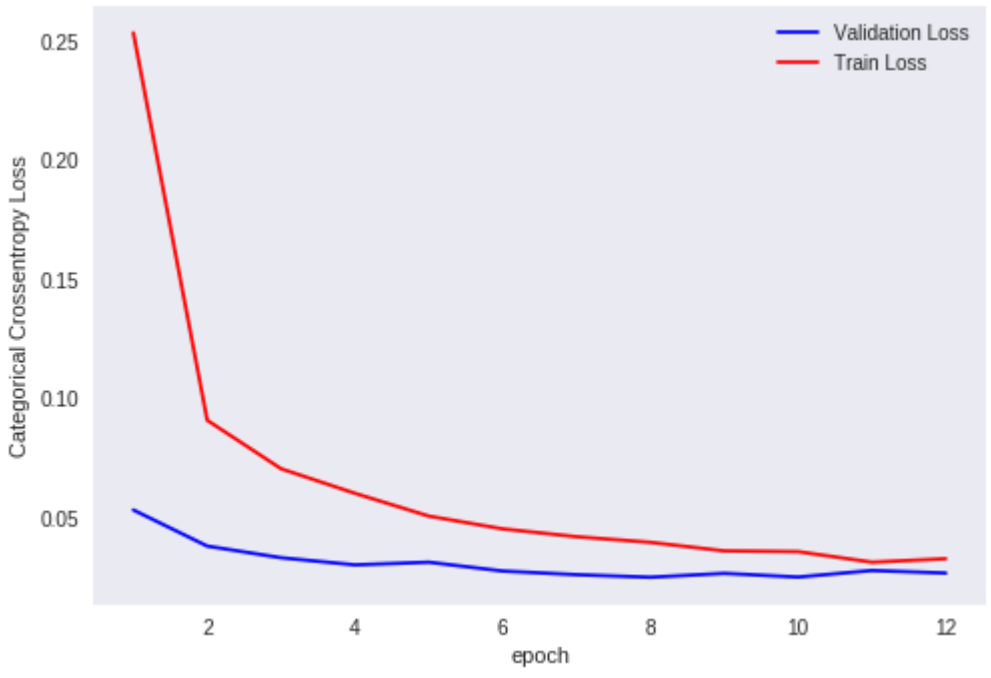
In [15]:
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.026414681418075996
Test accuracy: 0.9916
```

In [16]:
```
# Plot weight distribution using violin plot
plot_weights(model)

warnings.filterwarnings(action='ignore', category=DataConversionWarning)

print()
print()

# Plot train and cross validation error
plot_train_cv_loss(single_CNN, epochs)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:588: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  kde_data = remove_na(group_data)
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:816: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  violin_data = remove_na(group_data)
```





Implementing Three layer CNN network architecture

In [22]:
```
model_3l = Sequential()
#First convolution layer
model_3l.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model_3l.add(Conv2D(64, (3, 3), activation='relu'))
#First Max-pooling layer
model_3l.add(MaxPooling2D(pool_size=(2, 2)))
model_3l.add(Dropout(0.50))


#Second convolution layer
model_3l.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model_3l.add(Conv2D(64, (3, 3), activation='relu'))
#Second Max-pooling layer
model_3l.add(MaxPooling2D(pool_size=(2, 2)))
model_3l.add(Dropout(0.60))

#Third convolution layer
model_3l.add(Conv2D(16, kernel_size=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model_3l.add(Conv2D(8, (1, 1), activation='relu'))
#Third Max-pooling layer
model_3l.add(MaxPooling2D(pool_size=(2, 2)))

model_3l.add(Flatten())
model_3l.add(Dense(128, activation='relu'))
model_3l.add(Dropout(0.5))
model_3l.add(Dense(num_classes, activation='softmax'))

model_3l.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 10, 10, 32)        18464
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 8, 8, 64)          18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 4, 4, 16)          1040
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 4, 4, 8)           136
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 2, 2, 8)           0
_________________________________________________________________
flatten_4 (Flatten)          (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               4224
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
=================================================================
Total params: 62,466
Trainable params: 62,466
Non-trainable params: 0
_________________________________________________________________
```

Training the 3 layer CNN model over train set

In [23]:
```
model_3l.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

CNN_3L=model_3l.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 14s 237us/step - loss: 0.9270 - acc: 0.6765 - val_loss:
0.1470 - val_acc: 0.9537
Epoch 2/12
60000/60000 [==============================] - 12s 207us/step - loss: 0.2573 - acc: 0.9243 - val_loss:
0.0782 - val_acc: 0.9761
Epoch 3/12
60000/60000 [==============================] - 12s 196us/step - loss: 0.1887 - acc: 0.9460 - val_loss:
0.0660 - val_acc: 0.9797
Epoch 4/12
60000/60000 [==============================] - 12s 195us/step - loss: 0.1529 - acc: 0.9556 - val_loss:
0.0561 - val_acc: 0.9831
Epoch 5/12
60000/60000 [==============================] - 12s 195us/step - loss: 0.1332 - acc: 0.9625 - val_loss:
0.0462 - val_acc: 0.9856
Epoch 6/12
60000/60000 [==============================] - 12s 195us/step - loss: 0.1222 - acc: 0.9647 - val_loss:
0.0478 - val_acc: 0.9849
Epoch 7/12
60000/60000 [==============================] - 12s 194us/step - loss: 0.1105 - acc: 0.9680 - val_loss:
0.0458 - val_acc: 0.9848
Epoch 8/12
60000/60000 [==============================] - 12s 195us/step - loss: 0.1039 - acc: 0.9701 - val_loss:
0.0388 - val_acc: 0.9863
Epoch 9/12
60000/60000 [==============================] - 12s 195us/step - loss: 0.0945 - acc: 0.9718 - val_loss:
0.0375 - val_acc: 0.9870
Epoch 10/12
60000/60000 [==============================] - 12s 195us/step - loss: 0.0907 - acc: 0.9735 - val_loss:
0.0339 - val_acc: 0.9885
Epoch 11/12
60000/60000 [==============================] - 12s 195us/step - loss: 0.0877 - acc: 0.9744 - val_loss:
0.0341 - val_acc: 0.9886
Epoch 12/12
60000/60000 [==============================] - 12s 194us/step - loss: 0.0820 - acc: 0.9763 - val_loss:
0.0330 - val_acc: 0.9890
```

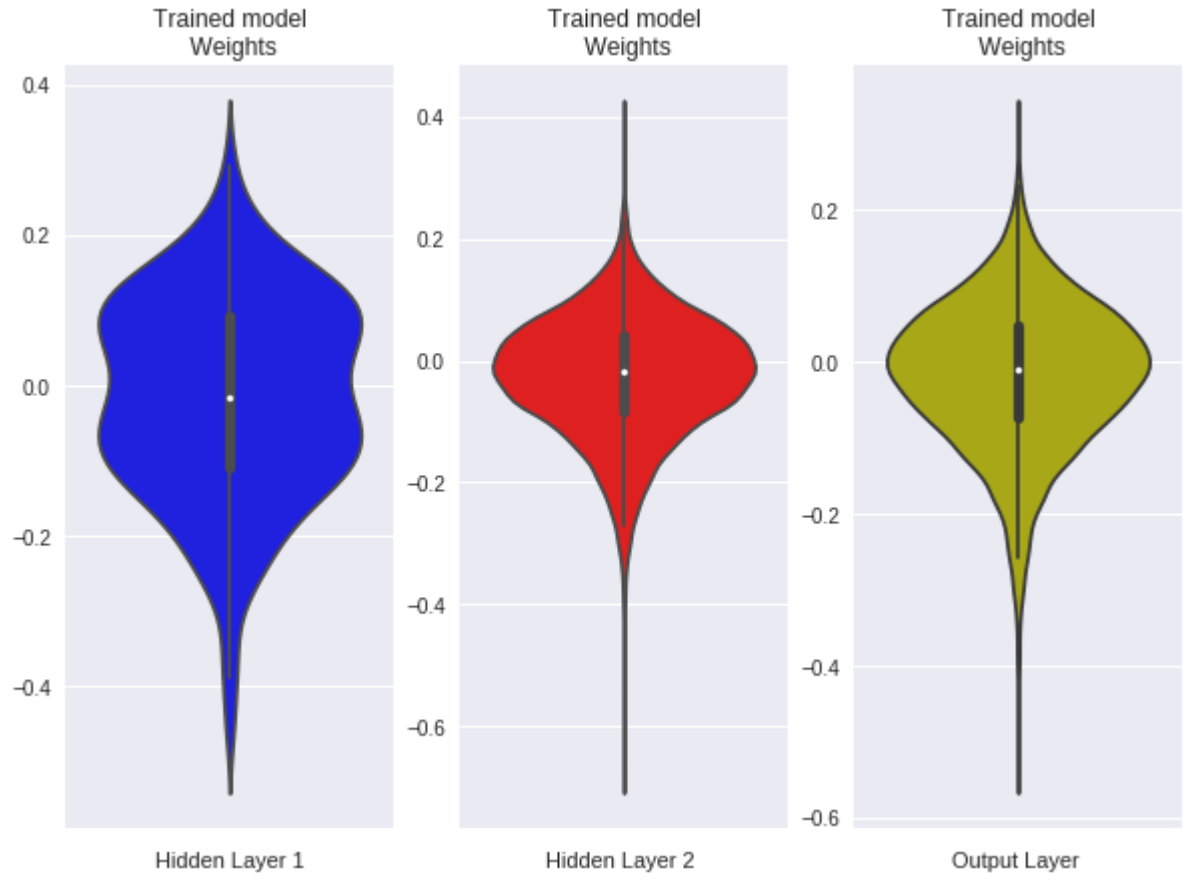
Testing the 3-layer over train set

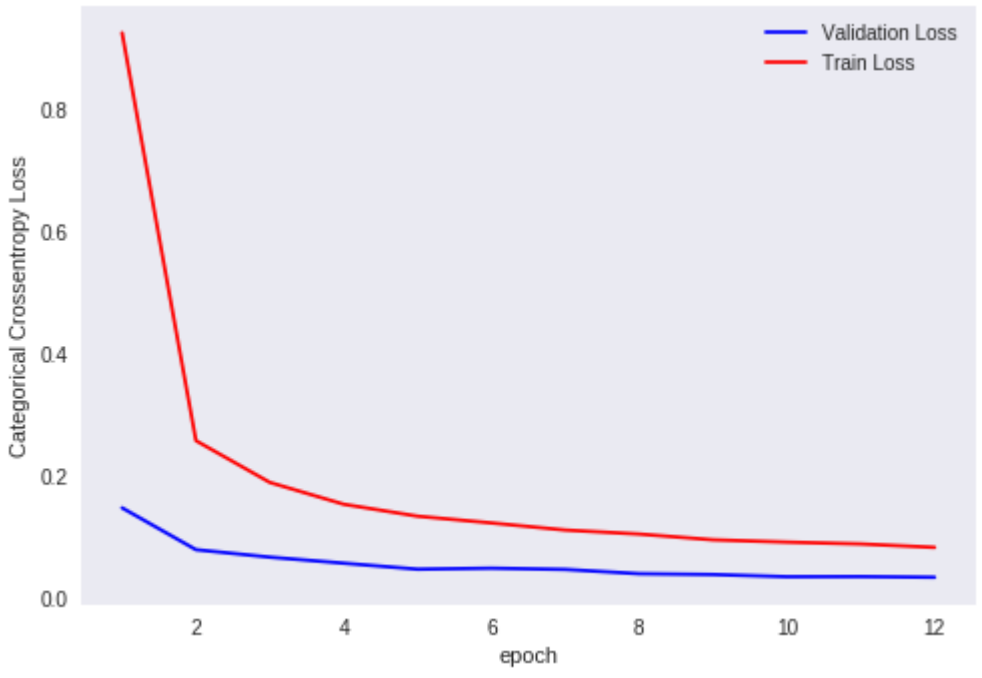
In [24]:
```
score_3l = model_3l.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score_3l[0])
print('Test accuracy:', score_3l[1])
```

```
Test loss: 0.03299151507723145
Test accuracy: 0.989
```

In [25]:
```
# Plot weight distribution using violin plot
plot_weights(model_3l)

warnings.filterwarnings(action='ignore', category=DataConversionWarning)

print()
print()

# Plot train and cross validation error
plot_train_cv_loss(CNN_3L, epochs)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:588: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  kde_data = remove_na(group_data)
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:816: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  violin_data = remove_na(group_data)
```





In [0]:

Implementing the Five layer CNN network architecture

In [26]:
```
model_5l = Sequential()
#First convolution layer
model_5l.add(Conv2D(32, kernel_size=(5, 5),   #padding is used for retaining the dimensions
                 activation='relu',
                 input_shape=input_shape))
model_5l.add(Conv2D(64, (3, 3), activation='relu',padding="same"))
#First Max-pooling layer
model_5l.add(MaxPooling2D(pool_size=(2, 2)))
model_5l.add(Dropout(0.5))

#Second convolution layer
model_5l.add(Conv2D(32, kernel_size=(3, 3),padding="same",    #padding is used for retaining the dimen
sions
                 activation='relu',
                 input_shape=input_shape))
model_5l.add(Conv2D(64, (3, 3), activation='relu',padding="same"))
#Second Max-pooling layer
model_5l.add(MaxPooling2D(pool_size=(1,1)))
model_5l.add(Dropout(0.60))

#Third convolution layer
model_5l.add(Conv2D(16, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model_5l.add(Conv2D(8, (2, 2), activation='relu'))
#Third Max-pooling layer
model_5l.add(MaxPooling2D(pool_size=(1, 1)))
model_5l.add(Dropout(0.5))

#Fourth convolution layer
model_5l.add(Conv2D(16, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model_5l.add(Conv2D(8, (2, 2), activation='relu',padding="same"))
#Fourth Max-pooling layer
model_5l.add(MaxPooling2D(pool_size=(1, 1)))
model_5l.add(Dropout(0.5))

#Fifth convolution layer
model_5l.add(Conv2D(16, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model_5l.add(Conv2D(8, (2, 2), activation='relu'))
#Fifth Max-pooling layer
model_5l.add(MaxPooling2D(pool_size=(1, 1)))
model_5l.add(Dropout(0.5))

model_5l.add(Flatten())
model_5l.add(Dense(128, activation='relu'))
model_5l.add(Dropout(0.5))
model_5l.add(Dense(num_classes, activation='softmax'))

model_5l.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_10 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 12, 12, 32)        18464
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_11 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 11, 11, 16)        4112
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 10, 10, 8)         520
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 10, 10, 8)         0
_________________________________________________________________
dropout_12 (Dropout)         (None, 10, 10, 8)         0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 6, 6, 16)          3216
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 6, 6, 8)           520
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 6, 6, 8)           0
_________________________________________________________________
dropout_13 (Dropout)         (None, 6, 6, 8)           0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 4, 4, 16)          1168
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 3, 3, 8)           520
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 3, 3, 8)           0
_________________________________________________________________
dropout_14 (Dropout)         (None, 3, 3, 8)           0
_________________________________________________________________
flatten_5 (Flatten)          (None, 72)                0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               9344
_________________________________________________________________
dropout_15 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 10)                1290
=================================================================
Total params: 76,978
Trainable params: 76,978
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

In [27]:
```
model_5l.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

CNN_5L=model_5l.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 18s 295us/step - loss: 1.5094 - acc: 0.4543 - val_loss:
0.4284 - val_acc: 0.8797
Epoch 2/12
60000/60000 [==============================] - 15s 251us/step - loss: 0.4994 - acc: 0.8532 - val_loss:
0.1936 - val_acc: 0.9470
Epoch 3/12
60000/60000 [==============================] - 15s 255us/step - loss: 0.3340 - acc: 0.9112 - val_loss:
0.1223 - val_acc: 0.9656
Epoch 4/12
60000/60000 [==============================] - 15s 252us/step - loss: 0.2694 - acc: 0.9299 - val_loss:
0.1111 - val_acc: 0.9686
Epoch 5/12
60000/60000 [==============================] - 15s 251us/step - loss: 0.2330 - acc: 0.9403 - val_loss:
0.0771 - val_acc: 0.9792
Epoch 6/12
60000/60000 [==============================] - 15s 252us/step - loss: 0.2001 - acc: 0.9477 - val_loss:
0.0819 - val_acc: 0.9768
Epoch 7/12
60000/60000 [==============================] - 15s 250us/step - loss: 0.1811 - acc: 0.9529 - val_loss:
0.0630 - val_acc: 0.9837
Epoch 8/12
60000/60000 [==============================] - 15s 251us/step - loss: 0.1653 - acc: 0.9580 - val_loss:
0.0760 - val_acc: 0.9797
Epoch 9/12
60000/60000 [==============================] - 15s 250us/step - loss: 0.1535 - acc: 0.9610 - val_loss:
0.0618 - val_acc: 0.9830
Epoch 10/12
60000/60000 [==============================] - 15s 251us/step - loss: 0.1468 - acc: 0.9629 - val_loss:
0.0514 - val_acc: 0.9847
Epoch 11/12
60000/60000 [==============================] - 15s 250us/step - loss: 0.1358 - acc: 0.9653 - val_loss:
0.0459 - val_acc: 0.9873
Epoch 12/12
60000/60000 [==============================] - 15s 251us/step - loss: 0.1295 - acc: 0.9669 - val_loss:
0.0481 - val_acc: 0.9851
```

In [0]:

In [28]:
```
score_5l = model_5l.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score_5l[0])
print('Test accuracy:', score_5l[1])
```

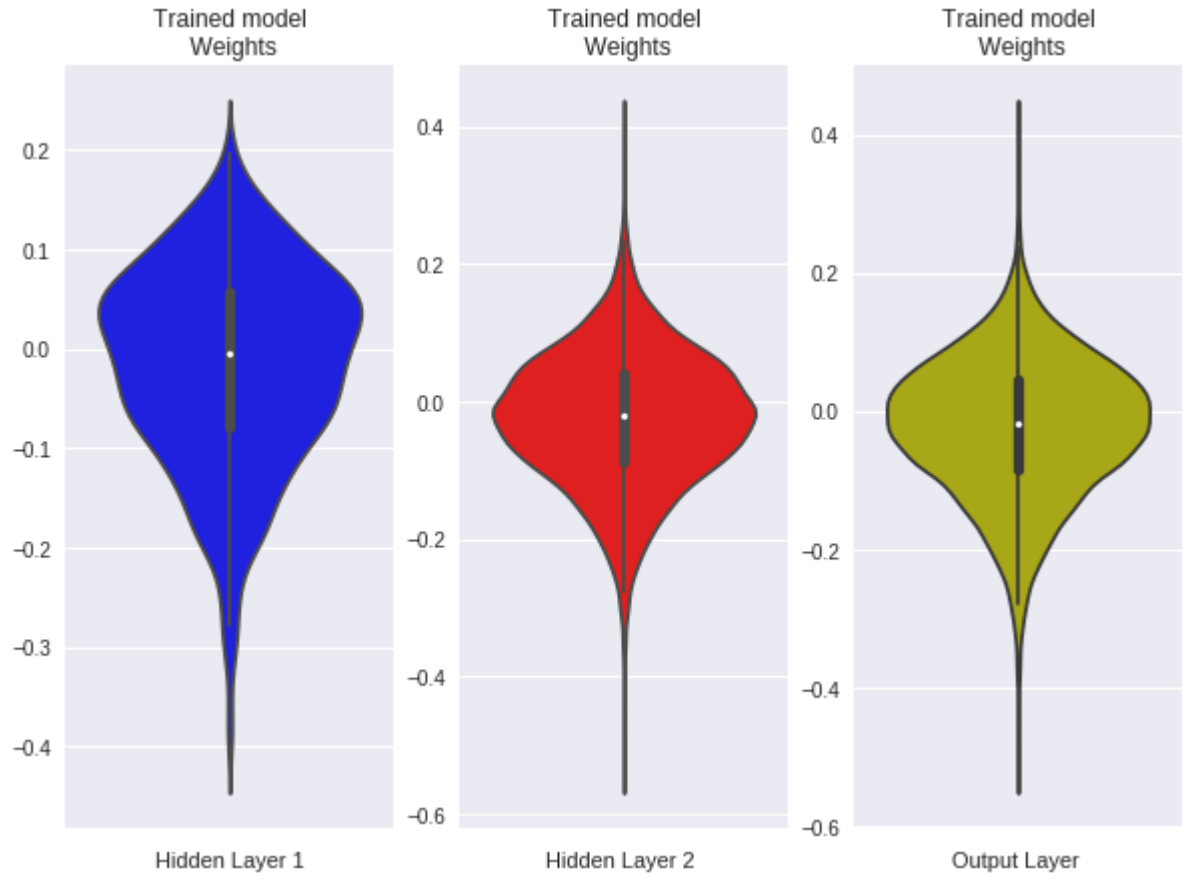
```
Test loss: 0.048075062668533064
Test accuracy: 0.9851
```

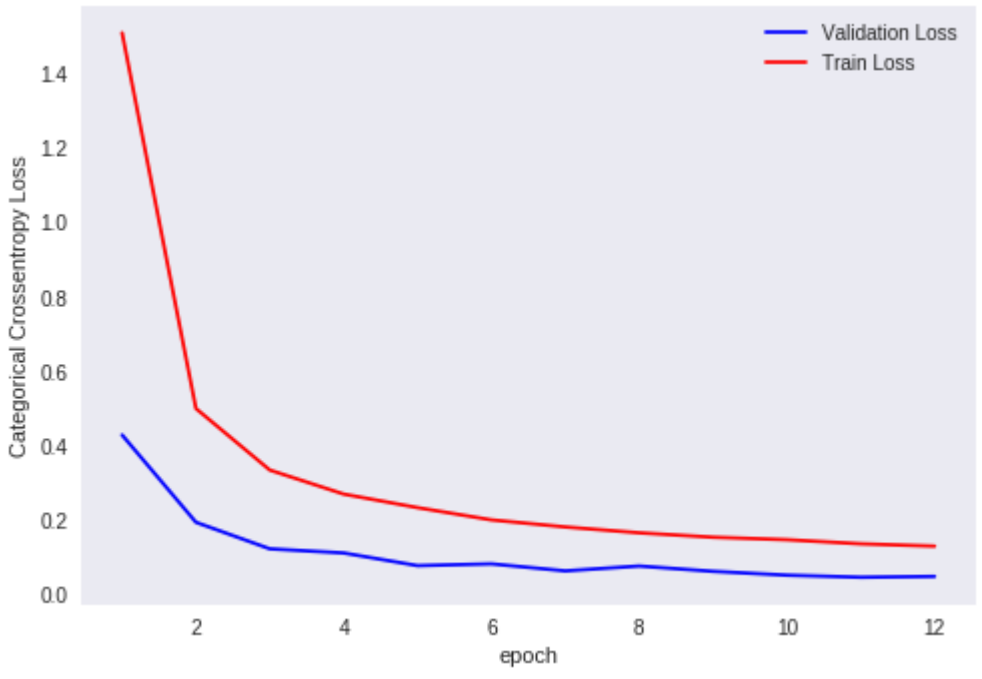
In [0]:

In [29]:
```
# Plot weight distribution using violin plot
plot_weights(model_5l)

warnings.filterwarnings(action='ignore', category=DataConversionWarning)

print()
print()
```

```
# Plot train and cross validation error
plot_train_cv_loss(CNN_5L, epochs)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:588: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  kde_data = remove_na(group_data)
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:816: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  violin_data = remove_na(group_data)
```





Implementinig the 7-layer CNN architecture

In [34]:
```
model_7l = Sequential()


#First convolution layer
model_7l.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))
model_7l.add(Conv2D(64, (3, 3), activation='relu'))
#First Max-pooling layer
model_7l.add(MaxPooling2D(pool_size=(2, 2)))
model_7l.add(Dropout(0.70))

#Second convolution layer
model_7l.add(Conv2D(32, kernel_size=(3, 3),padding="same",    #padding is used for retaining the dimen
sions
                 activation='relu',
                 input_shape=input_shape))
model_7l.add(Conv2D(64, (3, 3), activation='relu',padding="same"))  #padding is used for retaining the
 dimensions
#Second Max-pooling layer
model_7l.add(MaxPooling2D(pool_size=(1,1)))
model_7l.add(Dropout(0.60))

#Third convolution layer
model_7l.add(Conv2D(16, kernel_size=(7, 7),padding="same",  #padding is used for retaining the dimensi
ons
                 activation='relu',
                 input_shape=input_shape))
model_7l.add(Conv2D(8, (5, 5), activation='relu',padding="same"))   #padding is used for retaining the
 dimensions
#Third Max-pooling layer
model_7l.add(MaxPooling2D(pool_size=(1, 1)))
model_7l.add(Dropout(0.80))

#Fourth convolution layer
model_7l.add(Conv2D(16, kernel_size=(3, 3),padding="same",    #padding is used for retaining the dimen
sions
                 activation='relu',
                 input_shape=input_shape))
model_7l.add(Conv2D(8, (2, 2), activation='relu',padding="same"))   #padding is used for retaining the
 dimensions
#Fourth Max-pooling layer
model_7l.add(MaxPooling2D(pool_size=(1, 1)))
model_7l.add(Dropout(0.50))

#Fifth convolution layer
model_7l.add(Conv2D(16, kernel_size=(3, 3),padding="same",  #padding is used for retaining the dimensi
ons
                 activation='relu',
                 input_shape=input_shape))
model_7l.add(Conv2D(8, (2, 2), activation='relu',padding="same"))  #padding is used for retaining the
 dimensions
#Fifth Max-pooling layer
model_7l.add(MaxPooling2D(pool_size=(1, 1)))
model_7l.add(Dropout(0.80))

#Sixth convolution layer
model_7l.add(Conv2D(16, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))
model_7l.add(Conv2D(8, (1, 1), activation='relu'))
#Sixth Max-pooling layer
model_7l.add(MaxPooling2D(pool_size=(1, 1)))
model_7l.add(Dropout(0.70))

#Seventh convolution layer
model_7l.add(Conv2D(16, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model_7l.add(Conv2D(8, (2, 2), activation='relu'))
#Sixth Max-pooling layer
model_7l.add(MaxPooling2D(pool_size=(1, 1)))
model_7l.add(Dropout(0.90))



model_7l.add(Flatten())
model_7l.add(Dense(128, activation='relu'))
model_7l.add(Dropout(0.7))
model_7l.add(Dense(num_classes, activation='softmax'))

model_7l.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_41 (Conv2D)           (None, 22, 22, 32)        1600
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 20, 20, 64)        18496
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_24 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 10, 10, 32)        18464
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 10, 10, 64)        18496
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_25 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 10, 10, 16)        50192
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 10, 10, 8)         3208
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 10, 10, 8)         0
_________________________________________________________________
dropout_26 (Dropout)         (None, 10, 10, 8)         0
_________________________________________________________________
conv2d_47 (Conv2D)           (None, 10, 10, 16)        1168
_________________________________________________________________
conv2d_48 (Conv2D)           (None, 10, 10, 8)         520
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 10, 10, 8)         0
_________________________________________________________________
dropout_27 (Dropout)         (None, 10, 10, 8)         0
_________________________________________________________________
conv2d_49 (Conv2D)           (None, 10, 10, 16)        1168
_________________________________________________________________
conv2d_50 (Conv2D)           (None, 10, 10, 8)         520
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 10, 10, 8)         0
_________________________________________________________________
dropout_28 (Dropout)         (None, 10, 10, 8)         0
_________________________________________________________________
conv2d_51 (Conv2D)           (None, 4, 4, 16)          6288
_________________________________________________________________
conv2d_52 (Conv2D)           (None, 4, 4, 8)           136
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 4, 4, 8)           0
_________________________________________________________________
dropout_29 (Dropout)         (None, 4, 4, 8)           0
_________________________________________________________________
conv2d_53 (Conv2D)           (None, 3, 3, 16)          528
_________________________________________________________________
conv2d_54 (Conv2D)           (None, 2, 2, 8)           520
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 2, 2, 8)           0
_________________________________________________________________
dropout_30 (Dropout)         (None, 2, 2, 8)           0
_________________________________________________________________
flatten_7 (Flatten)          (None, 32)                0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               4224
_________________________________________________________________
dropout_31 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                1290
=================================================================
Total params: 126,818
Trainable params: 126,818
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

In [35]:
```
model_7l.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

cnn_7l=model_7l.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 20s 331us/step - loss: 2.3018 - acc: 0.1109 - val_loss:
2.3011 - val_acc: 0.1135
Epoch 2/12
60000/60000 [==============================] - 16s 268us/step - loss: 2.3015 - acc: 0.1123 - val_loss:
2.3011 - val_acc: 0.1135
Epoch 3/12
60000/60000 [==============================] - 16s 267us/step - loss: 2.3014 - acc: 0.1124 - val_loss:
2.3010 - val_acc: 0.1135
Epoch 4/12
60000/60000 [==============================] - 16s 266us/step - loss: 2.3013 - acc: 0.1124 - val_loss:
2.3010 - val_acc: 0.1135
Epoch 5/12
60000/60000 [==============================] - 16s 267us/step - loss: 2.3012 - acc: 0.1124 - val_loss:
2.3011 - val_acc: 0.1135
Epoch 6/12
60000/60000 [==============================] - 16s 271us/step - loss: 2.3013 - acc: 0.1124 - val_loss:
2.3011 - val_acc: 0.1135
Epoch 7/12
60000/60000 [==============================] - 16s 266us/step - loss: 2.3013 - acc: 0.1124 - val_loss:
2.3010 - val_acc: 0.1135
Epoch 8/12
60000/60000 [==============================] - 16s 266us/step - loss: 2.3013 - acc: 0.1124 - val_loss:
2.3010 - val_acc: 0.1135
Epoch 9/12
60000/60000 [==============================] - 16s 266us/step - loss: 2.3013 - acc: 0.1124 - val_loss:
2.3010 - val_acc: 0.1135
Epoch 10/12
60000/60000 [==============================] - 16s 266us/step - loss: 2.3012 - acc: 0.1124 - val_loss:
2.3010 - val_acc: 0.1135
Epoch 11/12
60000/60000 [==============================] - 16s 267us/step - loss: 2.3013 - acc: 0.1124 - val_loss:
2.3011 - val_acc: 0.1135
Epoch 12/12
60000/60000 [==============================] - 16s 266us/step - loss: 2.3013 - acc: 0.1124 - val_loss:
2.3010 - val_acc: 0.1135
```

In [0]:

In [36]:
```
score_7l = model_7l.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score_7l[0])
print('Test accuracy:', score_7l[1])
```

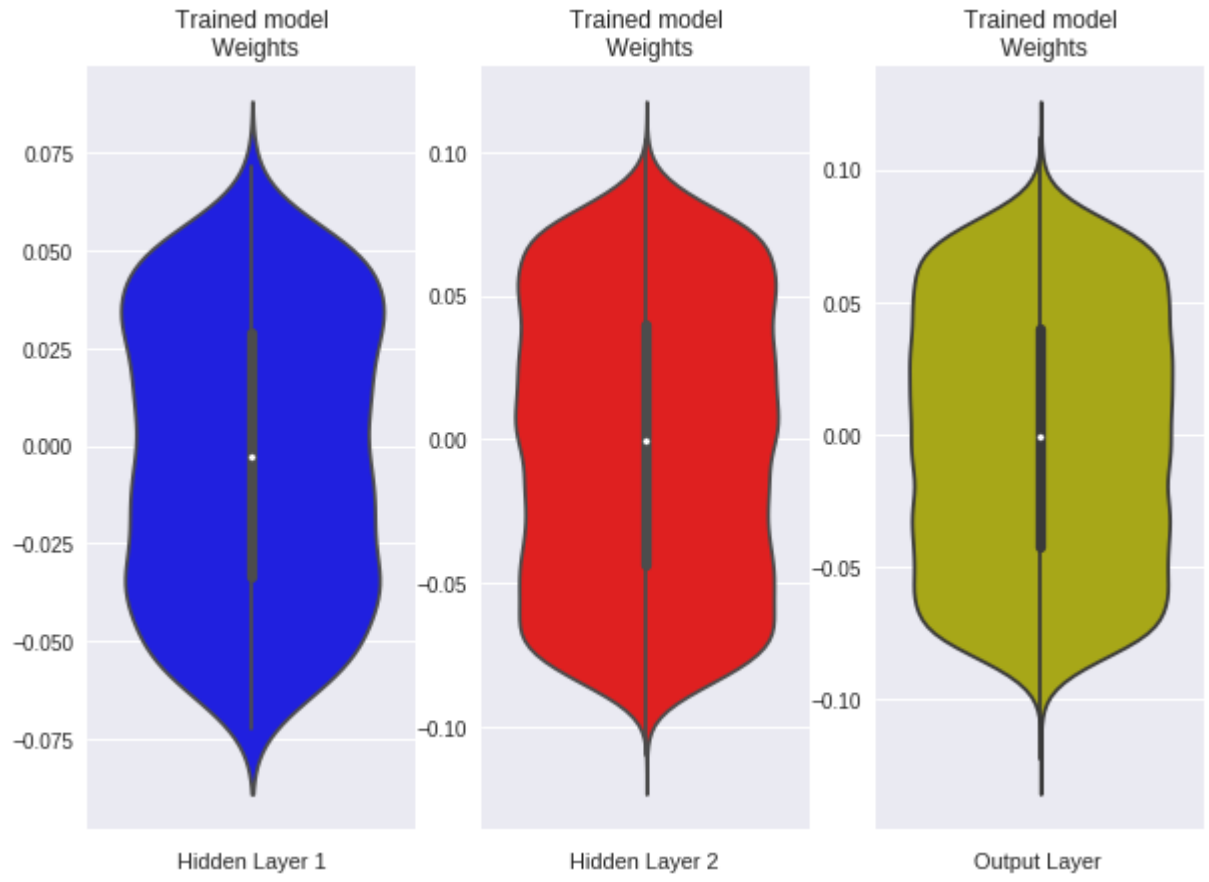
```
Test loss: 2.301028175354004
Test accuracy: 0.1135
```

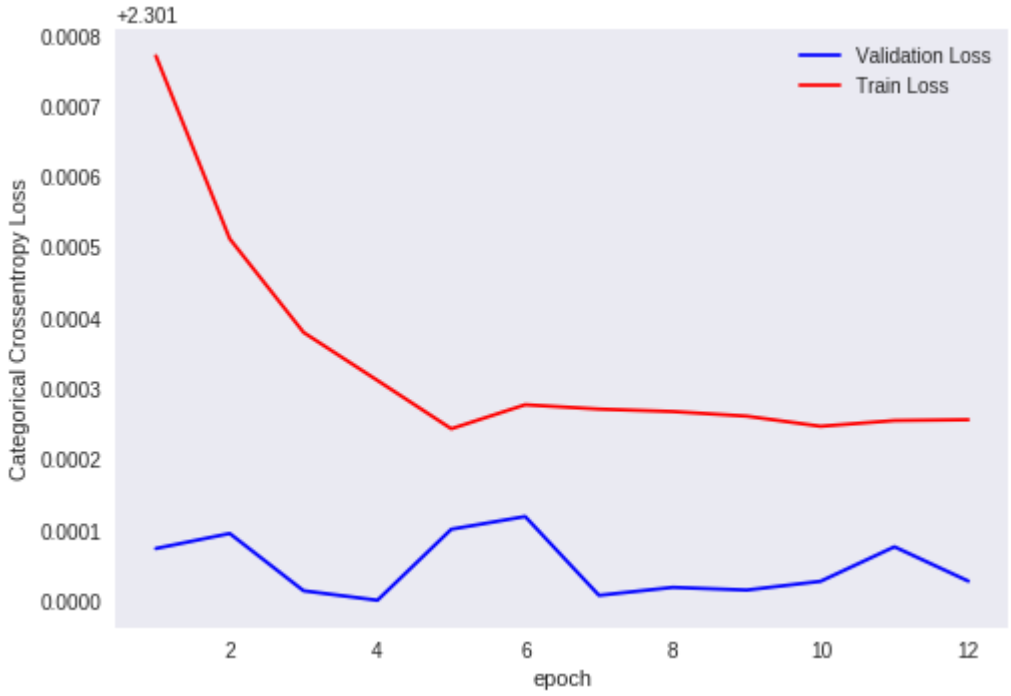
In [0]:

In [37]:
```
# Plot weight distribution using violin plot
plot_weights(model_7l)

warnings.filterwarnings(action='ignore', category=DataConversionWarning)

print()
print()
```

```
# Plot train and cross validation error
plot_train_cv_loss(cnn_7l, epochs)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:588: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  kde_data = remove_na(group_data)
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:816: FutureWarning: remove_na is depreca
ted and is a private function. Do not use.
  violin_data = remove_na(group_data)
```





OBSERVATIONS

- While training a simple convolution neural network with only one layer the accuracy was about 99.06% and the test loss was 2%, which was very good
- The test accuracy with three & five layer convolution network was around 98.99% and 98.62% and the test losses were 3% and 5.3%
- The seven layered convolution neural network performed worse by an accuracy of 56.56% and have highest loss value of 0.94 which is very bad for a model
- The model's performance can be increased by hyperparameter tuning.

In [0]: