In [0]:

```
import keras
import numpy as NP
from keras.datasets import mnist
```

In [0]:

```
(X_TRAIN,Y_TRAIN),(X_TEST,Y_TEST) = mnist.load_data()
```

In [12]:

```
X_TRAIN.shape,X_TEST.shape
```

Out[12]:

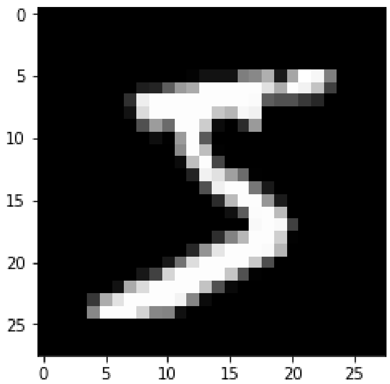
```
((60000, 28, 28), (10000, 28, 28))
```

In [13]:

```
import matplotlib.pyplot as plt
plt.imshow(X_TRAIN[0],'gray')
```

Out[13]:

```
<matplotlib.image.AxesImage at 0x7ff2b4e3ff60>
```



In [0]:

```
X_TRAIN = NP.reshape(X_TRAIN,(60000,784))
X_TEST = NP.reshape(X_TEST,(10000,784))
```

In [15]:

```
X_TRAIN.shape,X_TEST.shape
```

Out[15]:

```
((60000, 784), (10000, 784))
```

In [0]:

```
X_TRAIN = X_TRAIN/255
X_TEST = X_TEST/255
```

In [0]:

```
Y_TEST = keras.utils.to_categorical(Y_TEST,num_classes=10,dtype='int32')
Y_TRAIN = keras.utils.to_categorical(Y_TRAIN,num_classes=10,dtype='int32')
```

In [0]:

```
Y_TEST[0]
```

Out[0]:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int32)
```

2 Hidden Layer MLP having no Dropout and Batch Normalization

Neuron in Hidden Layers = [200-100]

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense,Activation
from keras.initializers import he_normal
```

In [0]:

```
MODEL = Sequential()
MODEL.add(Dense(200,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(Dense(10,activation='softmax'))
```

In [0]:

```
MODEL.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_105 (Dense)            (None, 200)               157000
_________________________________________________________________
dense_106 (Dense)            (None, 100)               20100
_________________________________________________________________
dense_107 (Dense)            (None, 10)                1010
=================================================================
Total params: 178,110
Trainable params: 178,110
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 8s 139us/step - loss: 0.2852 - acc: 0.9175 -
val_loss: 0.1412 - val_acc: 0.9572
Epoch 2/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.1084 - acc: 0.9683 -
val_loss: 0.0978 - val_acc: 0.9697
Epoch 3/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0726 - acc: 0.9778 -
val_loss: 0.0760 - val_acc: 0.9765
Epoch 4/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.0563 - acc: 0.9829 -
val_loss: 0.0778 - val_acc: 0.9763
Epoch 5/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.0413 - acc: 0.9871 -
val_loss: 0.0758 - val_acc: 0.9761
Epoch 6/30
```

```
Epoch 6/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.0326 - acc: 0.9897 -
val_loss: 0.0732 - val_acc: 0.9767
Epoch 7/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0258 - acc: 0.9919 -
val_loss: 0.0758 - val_acc: 0.9782
Epoch 8/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0212 - acc: 0.9935 -
val_loss: 0.0755 - val_acc: 0.9794
Epoch 9/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0173 - acc: 0.9943 -
val_loss: 0.0756 - val_acc: 0.9791
Epoch 10/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0141 - acc: 0.9957 -
val_loss: 0.0905 - val_acc: 0.9775
Epoch 11/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0137 - acc: 0.9955 -
val_loss: 0.0900 - val_acc: 0.9777
Epoch 12/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0109 - acc: 0.9963 -
val_loss: 0.0832 - val_acc: 0.9789
Epoch 13/30
60000/60000 [==============================] - 4s 58us/step - loss: 0.0106 - acc: 0.9966 -
val_loss: 0.0881 - val_acc: 0.9788
Epoch 14/30
60000/60000 [==============================] - 4s 58us/step - loss: 0.0073 - acc: 0.9978 -
val_loss: 0.0812 - val_acc: 0.9795
Epoch 15/30
60000/60000 [==============================] - 4s 59us/step - loss: 0.0124 - acc: 0.9959 -
val_loss: 0.0967 - val_acc: 0.9779
Epoch 16/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0111 - acc: 0.9961 -
val_loss: 0.0994 - val_acc: 0.9774
Epoch 17/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0047 - acc: 0.9986 -
val_loss: 0.0889 - val_acc: 0.9795
Epoch 18/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.0094 - acc: 0.9970 -
val_loss: 0.1188 - val_acc: 0.9766
Epoch 19/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0083 - acc: 0.9972 -
val_loss: 0.0855 - val_acc: 0.9801
Epoch 20/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0048 - acc: 0.9985 -
val_loss: 0.0962 - val_acc: 0.9793
Epoch 21/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0107 - acc: 0.9967 -
val_loss: 0.0912 - val_acc: 0.9793
Epoch 22/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.0048 - acc: 0.9987 -
val_loss: 0.0886 - val_acc: 0.9817
Epoch 23/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0077 - acc: 0.9975 -
val_loss: 0.1033 - val_acc: 0.9804
Epoch 24/30
60000/60000 [==============================] - 4s 61us/step - loss: 0.0072 - acc: 0.9979 -
val_loss: 0.0945 - val_acc: 0.9812
Epoch 25/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0025 - acc: 0.9993 -
val_loss: 0.0975 - val_acc: 0.9814
Epoch 26/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0069 - acc: 0.9977 -
val_loss: 0.1216 - val_acc: 0.9762
Epoch 27/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0111 - acc: 0.9965 -
val_loss: 0.1162 - val_acc: 0.9775
Epoch 28/30
60000/60000 [==============================] - 4s 60us/step - loss: 0.0045 - acc: 0.9986 -
val_loss: 0.1005 - val_acc: 0.9808
Epoch 29/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.0016 - acc: 0.9997 -
val_loss: 0.0959 - val_acc: 0.9814
Epoch 30/30
60000/60000 [==============================] - 4s 61us/step - loss: 5.0075e-04 - acc: 1.0000 - val
_loss: 0.0955 - val_acc: 0.9820
```

In [0]:

```
print(FINAL_MODEL.history.keys())
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
```

In [0]:

```
TEST_ACCURACY = []
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

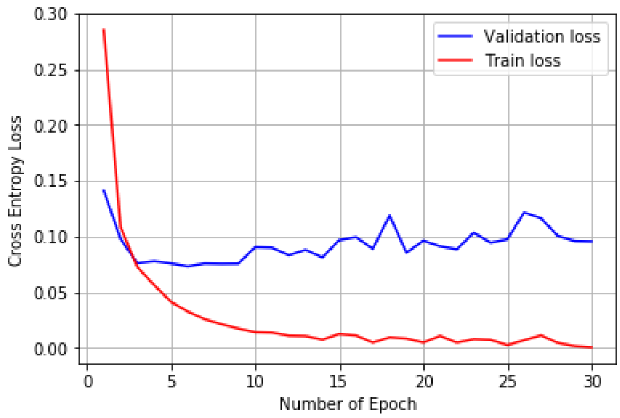
```
Test score: 0.09554963889349678
Test accuracy: 0.982
```

In [0]:

```
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 200)
(200,)
(200, 100)
(100,)
(100, 10)
(10,)
```

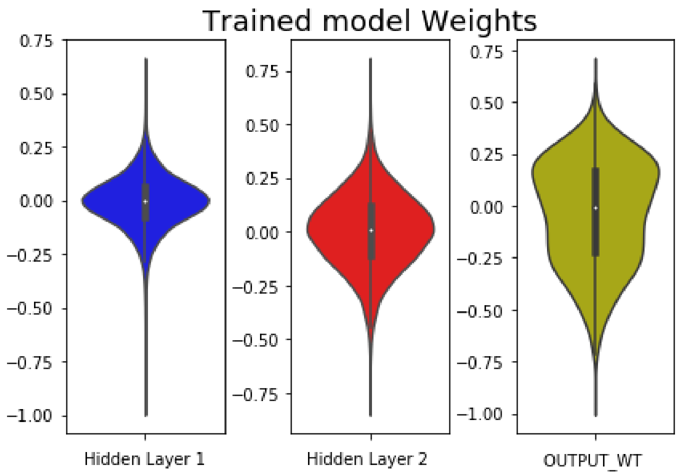
In [0]:

```
import seaborn as SNS
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
```

```
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[4].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3) = plt.subplots(nrows=1, ncols=3)
fig.tight_layout()

plt.subplot(1, 3, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 3, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 3, 3)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



2 Hidden Layer MLP + Batch Normalization

Neuron in Hidden Layers = [200-100]

In [0]:

```
from keras.layers.normalization import BatchNormalization
MODEL = Sequential()
MODEL.add(Dense(200,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dense(10,activation='softmax'))
print(MODEL.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_108 (Dense)            (None, 200)               157000    
_________________________________________________________________
batch_normalization_44 (Batc (None, 200)               800       
_________________________________________________________________
dense_109 (Dense)            (None, 100)               20100     
_________________________________________________________________
batch_normalization_45 (Batc (None, 100)               400       
_________________________________________________________________
dense_110 (Dense)            (None, 10)                1010      
=================================================================
Total params: 179,310
```

```
Total params: 179,310
Trainable params: 178,710
Non-trainable params: 600
_________________________________________________________________
None
```

In [0]:

```
MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 9s 155us/step - loss: 0.2312 - acc: 0.9321 -
val_loss: 0.1161 - val_acc: 0.9632
Epoch 2/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0873 - acc: 0.9742 -
val_loss: 0.0863 - val_acc: 0.9731
Epoch 3/30
60000/60000 [==============================] - 5s 81us/step - loss: 0.0582 - acc: 0.9821 -
val_loss: 0.0835 - val_acc: 0.9754
Epoch 4/30
60000/60000 [==============================] - 5s 79us/step - loss: 0.0440 - acc: 0.9862 -
val_loss: 0.0796 - val_acc: 0.9752
Epoch 5/30
60000/60000 [==============================] - 5s 79us/step - loss: 0.0339 - acc: 0.9890 -
val_loss: 0.0778 - val_acc: 0.9783
Epoch 6/30
60000/60000 [==============================] - 5s 79us/step - loss: 0.0247 - acc: 0.9922 -
val_loss: 0.0787 - val_acc: 0.9752
Epoch 7/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0190 - acc: 0.9941 -
val_loss: 0.0781 - val_acc: 0.9784
Epoch 8/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0193 - acc: 0.9937 -
val_loss: 0.0778 - val_acc: 0.9783
Epoch 9/30
60000/60000 [==============================] - 5s 79us/step - loss: 0.0202 - acc: 0.9935 -
val_loss: 0.0778 - val_acc: 0.9773
Epoch 10/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0132 - acc: 0.9956 -
val_loss: 0.0714 - val_acc: 0.9801
Epoch 11/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0134 - acc: 0.9954 -
val_loss: 0.0787 - val_acc: 0.9778
Epoch 12/30
60000/60000 [==============================] - 5s 79us/step - loss: 0.0131 - acc: 0.9957 -
val_loss: 0.0837 - val_acc: 0.9780
Epoch 13/30
60000/60000 [==============================] - 5s 82us/step - loss: 0.0095 - acc: 0.9972 -
val_loss: 0.0678 - val_acc: 0.9812
Epoch 14/30
60000/60000 [==============================] - 5s 82us/step - loss: 0.0109 - acc: 0.9966 -
val_loss: 0.0845 - val_acc: 0.9781
Epoch 15/30
60000/60000 [==============================] - 5s 79us/step - loss: 0.0097 - acc: 0.9967 -
val_loss: 0.0712 - val_acc: 0.9808
Epoch 16/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0087 - acc: 0.9972 -
val_loss: 0.0808 - val_acc: 0.9801
Epoch 17/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0080 - acc: 0.9974 -
val_loss: 0.0758 - val_acc: 0.9828
Epoch 18/30
60000/60000 [==============================] - 5s 79us/step - loss: 0.0065 - acc: 0.9980 -
val_loss: 0.0773 - val_acc: 0.9803
Epoch 19/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0095 - acc: 0.9968 -
val_loss: 0.0961 - val_acc: 0.9784
Epoch 20/30
60000/60000 [==============================] - 5s 83us/step - loss: 0.0094 - acc: 0.9968 -
val_loss: 0.0844 - val_acc: 0.9806
Epoch 21/30
60000/60000 [==============================] - 5s 83us/step - loss: 0.0072 - acc: 0.9976 -
val_loss: 0.0797 - val_acc: 0.9821
```

```
Epoch 22/30
60000/60000 [==============================] - 5s 82us/step - loss: 0.0051 - acc: 0.9984 -
val_loss: 0.0769 - val_acc: 0.9824
Epoch 23/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0053 - acc: 0.9983 -
val_loss: 0.0820 - val_acc: 0.9815
Epoch 24/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0057 - acc: 0.9982 -
val_loss: 0.0857 - val_acc: 0.9803
Epoch 25/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0081 - acc: 0.9972 -
val_loss: 0.0894 - val_acc: 0.9806
Epoch 26/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0064 - acc: 0.9978 -
val_loss: 0.0945 - val_acc: 0.9793
Epoch 27/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0073 - acc: 0.9975 -
val_loss: 0.0784 - val_acc: 0.9814
Epoch 28/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0037 - acc: 0.9988 -
val_loss: 0.0821 - val_acc: 0.9827
Epoch 29/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0034 - acc: 0.9989 -
val_loss: 0.0861 - val_acc: 0.9820
Epoch 30/30
60000/60000 [==============================] - 5s 80us/step - loss: 0.0075 - acc: 0.9976 -
val_loss: 0.0962 - val_acc: 0.9791
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

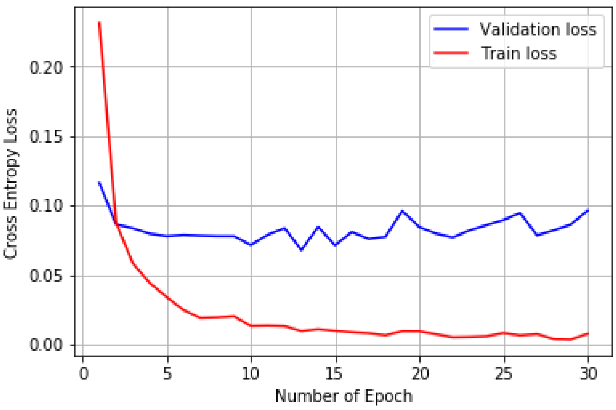
```
Test score: 0.09618185498487473
Test accuracy: 0.9791
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

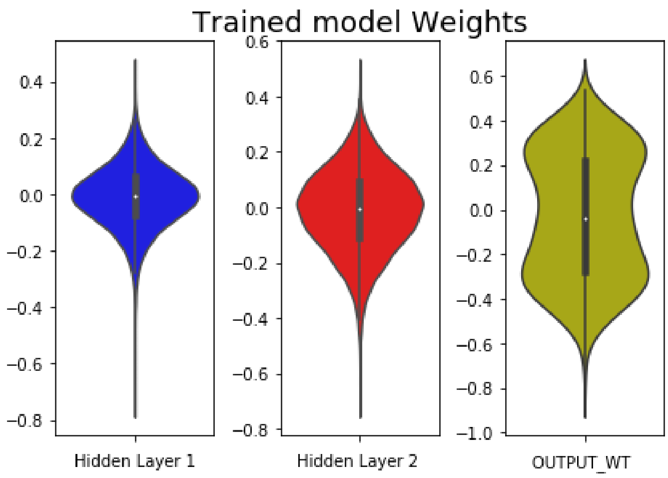
```
(784, 200)
(200,)
(200,)
(200,)
(200,)
(200,)
(200, 100)
(100,)
(100,)
(100,)
(100,)
(100,)
(100, 10)
(10,)
```

In [0]:

```
import seaborn as SNS
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[12].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3) = plt.subplots(nrows=1, ncols=3)
fig.tight_layout()

plt.subplot(1, 3, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 3, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 3, 3)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



2 Hidden Layer MLP + Dropout + Batch Normalization

Neuron in Hidden Layers = [200-100]

In [0]:

```
from keras.layers import Dropout

MODEL = Sequential()
MODEL.add(Dense(200,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))
MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))
MODEL.add(Dense(10,activation='softmax'))
print(MODEL.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_111 (Dense)            (None, 200)               157000
_________________________________________________________________
batch_normalization_46 (Batc (None, 200)               800
_________________________________________________________________
dropout_32 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_112 (Dense)            (None, 100)               20100
_________________________________________________________________
batch_normalization_47 (Batc (None, 100)               400
_________________________________________________________________
dropout_33 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_113 (Dense)            (None, 10)                1010
=================================================================
Total params: 179,310
Trainable params: 178,710
Non-trainable params: 600
_________________________________________________________________
None
```

In [0]:

```
MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.6045 - acc: 0.8177 - val_l
oss: 0.1930 - val_acc: 0.9402
Epoch 2/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.2957 - acc: 0.9125 -
val_loss: 0.1492 - val_acc: 0.9525
Epoch 3/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.2302 - acc: 0.9322 -
val_loss: 0.1180 - val_acc: 0.9647
Epoch 4/30
60000/60000 [==============================] - 6s 94us/step - loss: 0.1973 - acc: 0.9419 -
val_loss: 0.1049 - val_acc: 0.9673
Epoch 5/30
60000/60000 [==============================] - 6s 94us/step - loss: 0.1739 - acc: 0.9481 -
val_loss: 0.0939 - val_acc: 0.9723
Epoch 6/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1608 - acc: 0.9519 -
val_loss: 0.0924 - val_acc: 0.9732
Epoch 7/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1507 - acc: 0.9545 -
val_loss: 0.0887 - val_acc: 0.9750
Epoch 8/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.1366 - acc: 0.9597 -
val_loss: 0.0844 - val_acc: 0.9744
Epoch 9/30
60000/60000 [==============================] - 6s 94us/step - loss: 0.1298 - acc: 0.9609 -
val_loss: 0.0763 - val_acc: 0.9783
Epoch 10/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.1238 - acc: 0.9626 -
val_loss: 0.0804 - val_acc: 0.9763
Epoch 11/30
```

```
60000/60000 [==============================] - 6s 93us/step - loss: 0.1203 - acc: 0.9642 -
val_loss: 0.0751 - val_acc: 0.9771
Epoch 12/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.1158 - acc: 0.9653 -
val_loss: 0.0732 - val_acc: 0.9783
Epoch 13/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.1078 - acc: 0.9665 -
val_loss: 0.0733 - val_acc: 0.9783
Epoch 14/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1037 - acc: 0.9680 -
val_loss: 0.0694 - val_acc: 0.9787
Epoch 15/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1014 - acc: 0.9693 -
val_loss: 0.0707 - val_acc: 0.9788
Epoch 16/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.1016 - acc: 0.9694 -
val_loss: 0.0701 - val_acc: 0.9785
Epoch 17/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.0920 - acc: 0.9715 -
val_loss: 0.0672 - val_acc: 0.9795
Epoch 18/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.0908 - acc: 0.9722 -
val_loss: 0.0698 - val_acc: 0.9796
Epoch 19/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.0911 - acc: 0.9723 -
val_loss: 0.0686 - val_acc: 0.9796
Epoch 20/30
60000/60000 [==============================] - 6s 94us/step - loss: 0.0875 - acc: 0.9729 -
val_loss: 0.0671 - val_acc: 0.9811
Epoch 21/30
60000/60000 [==============================] - 6s 94us/step - loss: 0.0865 - acc: 0.9731 -
val_loss: 0.0649 - val_acc: 0.9817
Epoch 22/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.0827 - acc: 0.9742 -
val_loss: 0.0653 - val_acc: 0.9803
Epoch 23/30
60000/60000 [==============================] - 6s 94us/step - loss: 0.0831 - acc: 0.9735 -
val_loss: 0.0663 - val_acc: 0.9799
Epoch 24/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.0794 - acc: 0.9753 -
val_loss: 0.0653 - val_acc: 0.9820
Epoch 25/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.0805 - acc: 0.9745 -
val_loss: 0.0629 - val_acc: 0.9811
Epoch 26/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.0758 - acc: 0.9763 -
val_loss: 0.0619 - val_acc: 0.9814
Epoch 27/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.0753 - acc: 0.9755 -
val_loss: 0.0588 - val_acc: 0.9829
Epoch 28/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.0766 - acc: 0.9752 -
val_loss: 0.0637 - val_acc: 0.9813
Epoch 29/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.0750 - acc: 0.9765 -
val_loss: 0.0639 - val_acc: 0.9823
Epoch 30/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.0728 - acc: 0.9766 -
val_loss: 0.0659 - val_acc: 0.9814
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```
Test score: 0.06590885129593661
Test accuracy: 0.9814
```

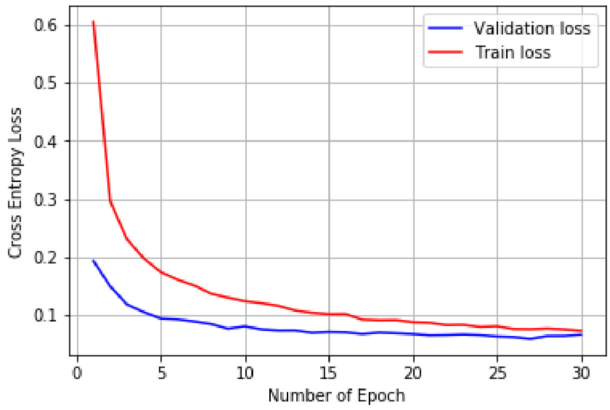
In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
```

```
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

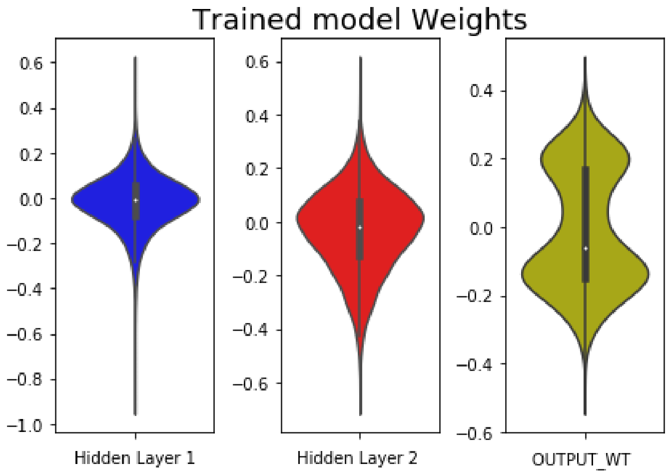
```
(784, 200)
(200,)
(200,)
(200,)
(200,)
(200,)
(200, 100)
(100,)
(100,)
(100,)
(100,)
(100,)
(100, 10)
(10,)
```

In [0]:

```
import seaborn as SNS

H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[12].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3) = plt.subplots(nrows=1, ncols=3)
fig.tight_layout()

plt.subplot(1, 3, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 3, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 3, 3)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



3 Hidden Layer MLP having no Dropout and Batch Normalization

Neuron in Hidden Layers = [200-100-50]

In [0]:

```
MODEL = Sequential()
MODEL.add(Dense(200,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(Dense(50,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(Dense(10,activation='softmax'))
MODEL.summary()
MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_114 (Dense)            (None, 200)               157000
_________________________________________________________________
dense_115 (Dense)            (None, 100)               20100
_________________________________________________________________
dense_116 (Dense)            (None, 50)                5050
_________________________________________________________________
dense_117 (Dense)            (None, 10)                510
=================================================================
Total params: 182,660
Trainable params: 182,660
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 9s 148us/step - loss: 0.2869 - acc: 0.9170 -
val_loss: 0.1396 - val_acc: 0.9596
Epoch 2/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.1113 - acc: 0.9666 -
val_loss: 0.0966 - val_acc: 0.9712
Epoch 3/30
60000/60000 [==============================] - 4s 73us/step - loss: 0.0748 - acc: 0.9770 -
val_loss: 0.0817 - val_acc: 0.9747
Epoch 4/30
60000/60000 [==============================] - 4s 73us/step - loss: 0.0538 - acc: 0.9834 -
val_loss: 0.0781 - val_acc: 0.9754
Epoch 5/30
60000/60000 [==============================] - 4s 70us/step - loss: 0.0427 - acc: 0.9860 -
```

```
val_loss: 0.0693 - val_acc: 0.9791
Epoch 6/30
60000/60000 [==============================] - 4s 71us/step - loss: 0.0336 - acc: 0.9889 -
val_loss: 0.0711 - val_acc: 0.9777
Epoch 7/30
60000/60000 [==============================] - 4s 71us/step - loss: 0.0260 - acc: 0.9916 -
val_loss: 0.0721 - val_acc: 0.9780
Epoch 8/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0216 - acc: 0.9930 -
val_loss: 0.0757 - val_acc: 0.9788
Epoch 9/30
60000/60000 [==============================] - 5s 76us/step - loss: 0.0212 - acc: 0.9929 -
val_loss: 0.0841 - val_acc: 0.9787
Epoch 10/30
60000/60000 [==============================] - 4s 74us/step - loss: 0.0215 - acc: 0.9928 -
val_loss: 0.0738 - val_acc: 0.9807
Epoch 11/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0132 - acc: 0.9957 -
val_loss: 0.0697 - val_acc: 0.9818
Epoch 12/30
60000/60000 [==============================] - 4s 70us/step - loss: 0.0135 - acc: 0.9956 -
val_loss: 0.0727 - val_acc: 0.9817
Epoch 13/30
60000/60000 [==============================] - 4s 71us/step - loss: 0.0111 - acc: 0.9967 -
val_loss: 0.0866 - val_acc: 0.9787
Epoch 14/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0127 - acc: 0.9957 -
val_loss: 0.0994 - val_acc: 0.9771
Epoch 15/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0160 - acc: 0.9945 -
val_loss: 0.0918 - val_acc: 0.9788
Epoch 16/30
60000/60000 [==============================] - 4s 71us/step - loss: 0.0110 - acc: 0.9960 -
val_loss: 0.0873 - val_acc: 0.9788
Epoch 17/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0064 - acc: 0.9980 -
val_loss: 0.0921 - val_acc: 0.9790
Epoch 18/30
60000/60000 [==============================] - 5s 75us/step - loss: 0.0099 - acc: 0.9965 -
val_loss: 0.0922 - val_acc: 0.9805
Epoch 19/30
60000/60000 [==============================] - 4s 74us/step - loss: 0.0113 - acc: 0.9962 -
val_loss: 0.0845 - val_acc: 0.9821
Epoch 20/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0062 - acc: 0.9980 -
val_loss: 0.1174 - val_acc: 0.9764
Epoch 21/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0110 - acc: 0.9963 -
val_loss: 0.1074 - val_acc: 0.9766
Epoch 22/30
60000/60000 [==============================] - 4s 70us/step - loss: 0.0092 - acc: 0.9969 -
val_loss: 0.1040 - val_acc: 0.9803
Epoch 23/30
60000/60000 [==============================] - 4s 70us/step - loss: 0.0061 - acc: 0.9978 -
val_loss: 0.1100 - val_acc: 0.9798
Epoch 24/30
60000/60000 [==============================] - 4s 73us/step - loss: 0.0095 - acc: 0.9969 -
val_loss: 0.0945 - val_acc: 0.9802
Epoch 25/30
60000/60000 [==============================] - 4s 74us/step - loss: 0.0091 - acc: 0.9970 -
val_loss: 0.1027 - val_acc: 0.9810
Epoch 26/30
60000/60000 [==============================] - 4s 73us/step - loss: 0.0067 - acc: 0.9978 -
val_loss: 0.0968 - val_acc: 0.9802
Epoch 27/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0023 - acc: 0.9992 -
val_loss: 0.0910 - val_acc: 0.9818
Epoch 28/30
60000/60000 [==============================] - 4s 73us/step - loss: 0.0112 - acc: 0.9965 -
val_loss: 0.1042 - val_acc: 0.9803
Epoch 29/30
60000/60000 [==============================] - 4s 74us/step - loss: 0.0078 - acc: 0.9975 -
val_loss: 0.1041 - val_acc: 0.9798
Epoch 30/30
60000/60000 [==============================] - 4s 72us/step - loss: 0.0074 - acc: 0.9977 -
val_loss: 0.0931 - val_acc: 0.9822
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

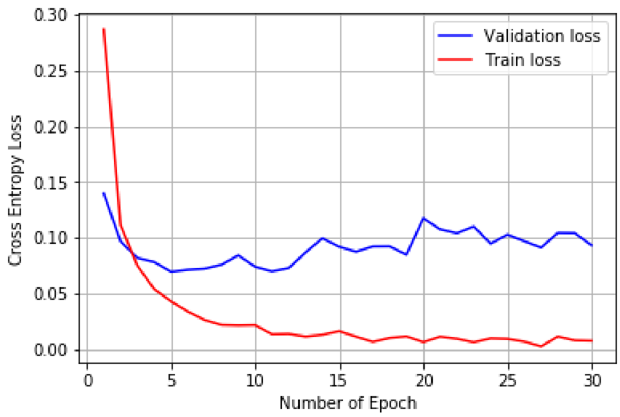
```
Test score: 0.09314377224932355
Test accuracy: 0.9822
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 200)
(200,)
(200, 100)
(100,)
(100, 50)
(50,)
(50, 10)
(10,)
```

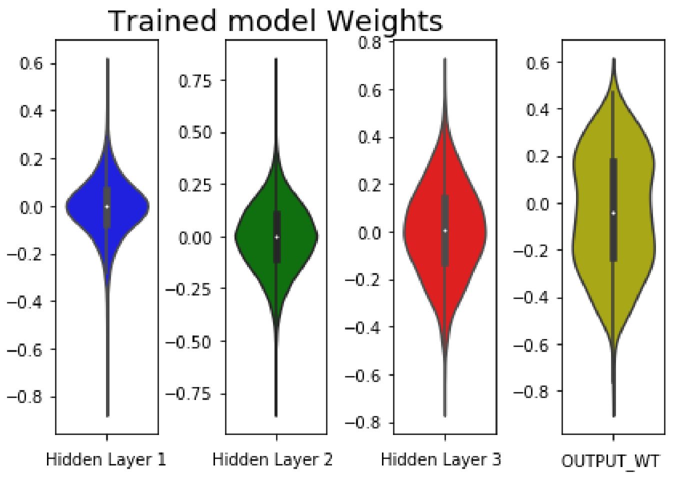
In [0]:

```
import seaborn as SNS
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
H3_WT = MODEL_WT[4].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[6].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
```

```
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='g')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='r')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



3 Hidden Layer MLP + Batch Normalization

Neuron in Hidden Layers = [200-100-50]

In [0]:

```
MODEL = Sequential()
MODEL.add(Dense(200,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dense(50,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dense(10,activation='softmax'))
MODEL.summary()
MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_118 (Dense)            (None, 200)               157000    
_________________________________________________________________
batch_normalization_48 (Batc (None, 200)               800       
_________________________________________________________________
dense_119 (Dense)            (None, 100)               20100     
_________________________________________________________________
batch_normalization_49 (Batc (None, 100)               400       
_________________________________________________________________
dense_120 (Dense)            (None, 50)                5050      
_________________________________________________________________
batch_normalization_50 (Batc (None, 50)                200       
```

```
_________________________________________________________________
dense_121 (Dense)            (None, 10)                510
=================================================================
Total params: 184,060
Trainable params: 183,360
Non-trainable params: 700
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.2666 - acc: 0.9238 - val_l
oss: 0.1230 - val_acc: 0.9646
Epoch 2/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0943 - acc: 0.9712 -
val_loss: 0.0955 - val_acc: 0.9695
Epoch 3/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0633 - acc: 0.9806 -
val_loss: 0.0909 - val_acc: 0.9718
Epoch 4/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0468 - acc: 0.9857 -
val_loss: 0.0844 - val_acc: 0.9748
Epoch 5/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0366 - acc: 0.9880 -
val_loss: 0.0781 - val_acc: 0.9769
Epoch 6/30
60000/60000 [==============================] - 5s 84us/step - loss: 0.0308 - acc: 0.9901 -
val_loss: 0.0800 - val_acc: 0.9763
Epoch 7/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0269 - acc: 0.9912 -
val_loss: 0.0842 - val_acc: 0.9755
Epoch 8/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0222 - acc: 0.9928 -
val_loss: 0.0780 - val_acc: 0.9771
Epoch 9/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0195 - acc: 0.9935 -
val_loss: 0.0744 - val_acc: 0.9799
Epoch 10/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0172 - acc: 0.9945 -
val_loss: 0.0812 - val_acc: 0.9782
Epoch 11/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0160 - acc: 0.9946 -
val_loss: 0.0804 - val_acc: 0.9777
Epoch 12/30
60000/60000 [==============================] - 5s 84us/step - loss: 0.0151 - acc: 0.9950 -
val_loss: 0.0748 - val_acc: 0.9799
Epoch 13/30
60000/60000 [==============================] - 5s 83us/step - loss: 0.0148 - acc: 0.9949 -
val_loss: 0.0845 - val_acc: 0.9785
Epoch 14/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0143 - acc: 0.9954 -
val_loss: 0.0803 - val_acc: 0.9785
Epoch 15/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0135 - acc: 0.9958 -
val_loss: 0.0837 - val_acc: 0.9807
Epoch 16/30
60000/60000 [==============================] - 5s 84us/step - loss: 0.0116 - acc: 0.9960 -
val_loss: 0.0988 - val_acc: 0.9771
Epoch 17/30
60000/60000 [==============================] - 5s 82us/step - loss: 0.0101 - acc: 0.9966 -
val_loss: 0.0955 - val_acc: 0.9772
Epoch 18/30
60000/60000 [==============================] - 5s 83us/step - loss: 0.0104 - acc: 0.9966 -
val_loss: 0.0920 - val_acc: 0.9771
Epoch 19/30
60000/60000 [==============================] - 5s 84us/step - loss: 0.0104 - acc: 0.9965 -
val_loss: 0.1030 - val_acc: 0.9746
Epoch 20/30
60000/60000 [==============================] - 5s 82us/step - loss: 0.0097 - acc: 0.9969 -
val_loss: 0.1020 - val_acc: 0.9761
Epoch 21/30
60000/60000 [==============================] - 5s 83us/step - loss: 0.0082 - acc: 0.9973 -
val_loss: 0.0879 - val_acc: 0.9788
Epoch 22/30
60000/60000 [==============================] - 5s 84us/step - loss: 0.0061 - acc: 0.9980 -
val_loss: 0.0828 - val_acc: 0.9803
Epoch 23/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0097 - acc: 0.9968 -
val_loss: 0.0865 - val_acc: 0.9794
```

```
val_loss: 0.0805 - val_acc: 0.9794
Epoch 24/30
60000/60000 [==============================] - 5s 84us/step - loss: 0.0116 - acc: 0.9960 -
val_loss: 0.0904 - val_acc: 0.9803
Epoch 25/30
60000/60000 [==============================] - 5s 83us/step - loss: 0.0079 - acc: 0.9973 -
val_loss: 0.0818 - val_acc: 0.9817
Epoch 26/30
60000/60000 [==============================] - 5s 84us/step - loss: 0.0041 - acc: 0.9988 -
val_loss: 0.0754 - val_acc: 0.9827
Epoch 27/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0061 - acc: 0.9980 -
val_loss: 0.0992 - val_acc: 0.9786
Epoch 28/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0100 - acc: 0.9966 -
val_loss: 0.0955 - val_acc: 0.9776
Epoch 29/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0087 - acc: 0.9970 -
val_loss: 0.0877 - val_acc: 0.9794
Epoch 30/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0053 - acc: 0.9980 -
val_loss: 0.0854 - val_acc: 0.9808
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

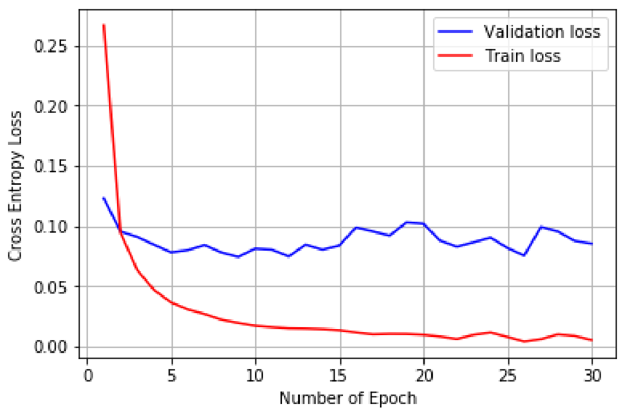
```
Test score: 0.0854192057800221
Test accuracy: 0.9808
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 200)
```

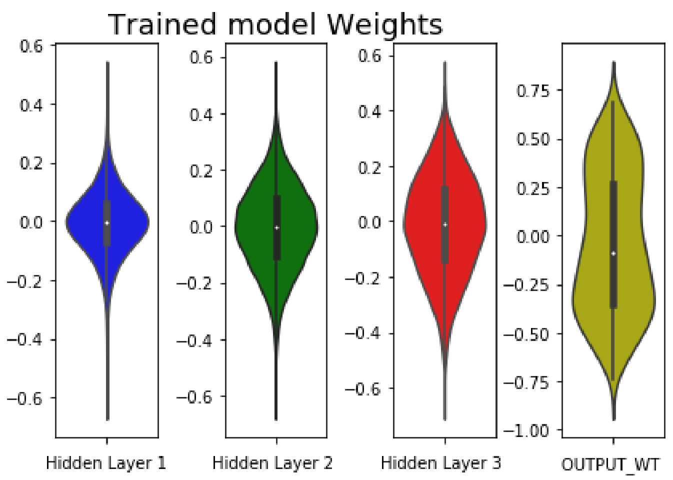
```
(200,)
(200,)
(200,)
(200,)
(200,)
(200, 100)
(100,)
(100,)
(100,)
(100,)
(100,)
(100, 50)
(50,)
(50,)
(50,)
(50,)
(50,)
(50, 10)
(10,)
```

In [0]:

```
import seaborn as SNS

H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[18].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='g')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='r')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



3 Hidden Layer MLP + Dropout + Batch Normalization

3 Hidden Layer MLP + Dropout + Batch Normalization

Neuron in Hidden Layers = [200-100-50]

In [0]:

```
MODEL = Sequential()
MODEL.add(Dense(200,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))
MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))
MODEL.add(Dense(50,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))
MODEL.add(Dense(10,activation='softmax'))
print(MODEL.summary())

MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_122 (Dense)            (None, 200)               157000
_________________________________________________________________
batch_normalization_51 (Batc (None, 200)               800
_________________________________________________________________
dropout_34 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_123 (Dense)            (None, 100)               20100
_________________________________________________________________
batch_normalization_52 (Batc (None, 100)               400
_________________________________________________________________
dropout_35 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_124 (Dense)            (None, 50)                5050
_________________________________________________________________
batch_normalization_53 (Batc (None, 50)                200
_________________________________________________________________
dropout_36 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_125 (Dense)            (None, 10)                510
=================================================================
Total params: 184,060
Trainable params: 183,360
Non-trainable params: 700
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 11s 185us/step - loss: 0.9296 - acc: 0.7140 - val_l
oss: 0.2417 - val_acc: 0.9281
Epoch 2/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.4077 - acc: 0.8867 -
val_loss: 0.1736 - val_acc: 0.9485
Epoch 3/30
60000/60000 [==============================] - 6s 100us/step - loss: 0.3159 - acc: 0.9131 -
val_loss: 0.1495 - val_acc: 0.9550
Epoch 4/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.2639 - acc: 0.9278 -
val_loss: 0.1309 - val_acc: 0.9618
Epoch 5/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.2367 - acc: 0.9367 -
val_loss: 0.1273 - val_acc: 0.9636
Epoch 6/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.2183 - acc: 0.9401 -
val_loss: 0.1107 - val_acc: 0.9688
Epoch 7/30
60000/60000 [==============================] - 6s 99us/step - loss: 0.1990 - acc: 0.9460 -
val_loss: 0.1044 - val_acc: 0.9705
Epoch 8/30
```

```
60000/60000 [==============================] - 6s 96us/step - loss: 0.1853 - acc: 0.9503 -
val_loss: 0.0978 - val_acc: 0.9718
Epoch 9/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1782 - acc: 0.9525 -
val_loss: 0.1002 - val_acc: 0.9711
Epoch 10/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.1713 - acc: 0.9544 -
val_loss: 0.0963 - val_acc: 0.9724
Epoch 11/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1642 - acc: 0.9559 -
val_loss: 0.0914 - val_acc: 0.9750
Epoch 12/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.1560 - acc: 0.9578 -
val_loss: 0.0860 - val_acc: 0.9761
Epoch 13/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.1488 - acc: 0.9592 -
val_loss: 0.0888 - val_acc: 0.9759
Epoch 14/30
60000/60000 [==============================] - 6s 98us/step - loss: 0.1420 - acc: 0.9611 -
val_loss: 0.0844 - val_acc: 0.9769
Epoch 15/30
60000/60000 [==============================] - 6s 101us/step - loss: 0.1365 - acc: 0.9624 -
val_loss: 0.0821 - val_acc: 0.9781
Epoch 16/30
60000/60000 [==============================] - 6s 100us/step - loss: 0.1312 - acc: 0.9642 -
val_loss: 0.0830 - val_acc: 0.9765
Epoch 17/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.1333 - acc: 0.9633 -
val_loss: 0.0831 - val_acc: 0.9783
Epoch 18/30
60000/60000 [==============================] - 6s 98us/step - loss: 0.1272 - acc: 0.9664 -
val_loss: 0.0796 - val_acc: 0.9787
Epoch 19/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1216 - acc: 0.9667 -
val_loss: 0.0829 - val_acc: 0.9781
Epoch 20/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1226 - acc: 0.9665 -
val_loss: 0.0738 - val_acc: 0.9792
Epoch 21/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.1175 - acc: 0.9678 -
val_loss: 0.0761 - val_acc: 0.9802
Epoch 22/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1202 - acc: 0.9674 -
val_loss: 0.0752 - val_acc: 0.9786
Epoch 23/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.1157 - acc: 0.9689 -
val_loss: 0.0761 - val_acc: 0.9794
Epoch 24/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.1115 - acc: 0.9694 -
val_loss: 0.0805 - val_acc: 0.9785
Epoch 25/30
60000/60000 [==============================] - 6s 94us/step - loss: 0.1064 - acc: 0.9702 -
val_loss: 0.0766 - val_acc: 0.9793
Epoch 26/30
60000/60000 [==============================] - 6s 95us/step - loss: 0.1052 - acc: 0.9706 -
val_loss: 0.0744 - val_acc: 0.9791
Epoch 27/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1052 - acc: 0.9715 -
val_loss: 0.0793 - val_acc: 0.9796
Epoch 28/30
60000/60000 [==============================] - 6s 97us/step - loss: 0.1050 - acc: 0.9706 -
val_loss: 0.0765 - val_acc: 0.9780
Epoch 29/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.1033 - acc: 0.9716 -
val_loss: 0.0733 - val_acc: 0.9806
Epoch 30/30
60000/60000 [==============================] - 6s 96us/step - loss: 0.0987 - acc: 0.9735 -
val_loss: 0.0737 - val_acc: 0.9811
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

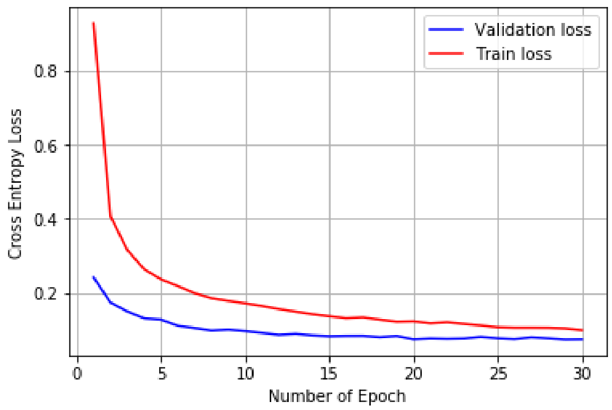
```
Test score: 0.07373714413648703
Test accuracy: 0.9811
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 200)
(200,)
(200,)
(200,)
(200,)
(200,)
(200, 100)
(100,)
(100,)
(100,)
(100,)
(100,)
(100, 50)
(50,)
(50,)
(50,)
(50,)
(50,)
(50, 10)
(10,)
```

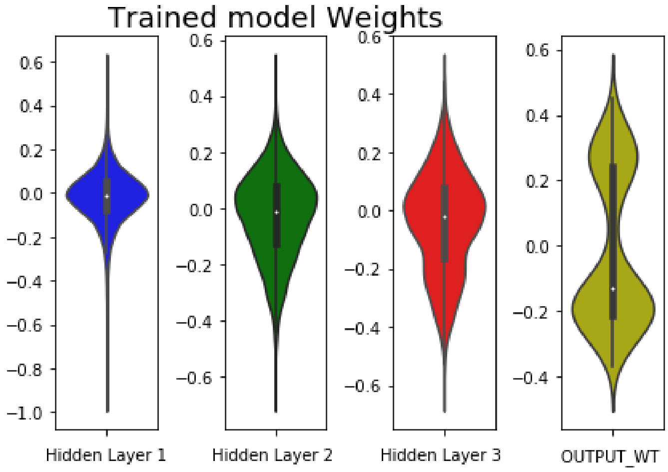
In [0]:

```
import seaborn as SNS

H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[18].flatten().reshape(-1,1)
```

```
fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='g')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='r')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



5 Hidden Layer MLP having no Dropout and Batch Normalization

Neuron in Hidden Layers = [400-300-200-100-50]

In [0]:

```
MODEL = Sequential()
MODEL.add(Dense(400,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))

MODEL.add(Dense(300,activation='relu',kernel_initializer=he_normal(seed=None)))

MODEL.add(Dense(200,activation='relu',kernel_initializer=he_normal(seed=None)))

MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))

MODEL.add(Dense(10,activation='relu',kernel_initializer=he_normal(seed=None)))

MODEL.add(Dense(10,activation='softmax'))
print(MODEL.summary())

MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
dense_126 (Dense)            (None, 400)               314000
_________________________________________________________________
dense_127 (Dense)            (None, 300)               120300
_________________________________________________________________
dense_128 (Dense)            (None, 200)               60200
_________________________________________________________________
dense_129 (Dense)            (None, 100)               20100
_________________________________________________________________
dense_130 (Dense)            (None, 10)                1010
_________________________________________________________________
dense_131 (Dense)            (None, 10)                110
=================================================================
Total params: 515,720
Trainable params: 515,720
Non-trainable params: 0
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 13s 218us/step - loss: 0.3083 - acc: 0.9053 - val_l
oss: 0.1471 - val_acc: 0.9545
Epoch 2/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0991 - acc: 0.9701 -
val_loss: 0.0847 - val_acc: 0.9750
Epoch 3/30
60000/60000 [==============================] - 8s 134us/step - loss: 0.0646 - acc: 0.9803 -
val_loss: 0.0889 - val_acc: 0.9731
Epoch 4/30
60000/60000 [==============================] - 8s 134us/step - loss: 0.0468 - acc: 0.9853 -
val_loss: 0.0791 - val_acc: 0.9784
Epoch 5/30
60000/60000 [==============================] - 8s 137us/step - loss: 0.0376 - acc: 0.9883 -
val_loss: 0.0903 - val_acc: 0.9757
Epoch 6/30
60000/60000 [==============================] - 8s 136us/step - loss: 0.0343 - acc: 0.9884 -
val_loss: 0.0850 - val_acc: 0.9775
Epoch 7/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0245 - acc: 0.9923 -
val_loss: 0.0994 - val_acc: 0.9764
Epoch 8/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0254 - acc: 0.9920 -
val_loss: 0.0758 - val_acc: 0.9829
Epoch 9/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0207 - acc: 0.9938 -
val_loss: 0.1040 - val_acc: 0.9769
Epoch 10/30
60000/60000 [==============================] - 8s 136us/step - loss: 0.0206 - acc: 0.9930 -
val_loss: 0.0899 - val_acc: 0.9786
Epoch 11/30
60000/60000 [==============================] - 8s 136us/step - loss: 0.0185 - acc: 0.9939 -
val_loss: 0.1180 - val_acc: 0.9734
Epoch 12/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0160 - acc: 0.9949 -
val_loss: 0.0938 - val_acc: 0.9794
Epoch 13/30
60000/60000 [==============================] - 8s 134us/step - loss: 0.0166 - acc: 0.9950 -
val_loss: 0.0831 - val_acc: 0.9803
Epoch 14/30
60000/60000 [==============================] - 8s 131us/step - loss: 0.0116 - acc: 0.9964 -
val_loss: 0.1125 - val_acc: 0.9759
Epoch 15/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0178 - acc: 0.9947 -
val_loss: 0.0864 - val_acc: 0.9800
Epoch 16/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0104 - acc: 0.9967 -
val_loss: 0.0843 - val_acc: 0.9830
Epoch 17/30
60000/60000 [==============================] - 8s 134us/step - loss: 0.0105 - acc: 0.9968 -
val_loss: 0.0959 - val_acc: 0.9811
Epoch 18/30
60000/60000 [==============================] - 8s 133us/step - loss: 0.0119 - acc: 0.9963 -
val_loss: 0.0906 - val_acc: 0.9811
Epoch 19/30
60000/60000 [==============================] - 8s 133us/step - loss: 0.0107 - acc: 0.9968 -
val_loss: 0.0953 - val_acc: 0.9828
Epoch 20/30
```

```
60000/60000 [==============================] - 8s 134us/step - loss: 0.0113 - acc: 0.9968 -
val_loss: 0.0928 - val_acc: 0.9810
Epoch 21/30
60000/60000 [==============================] - 8s 135us/step - loss: 0.0120 - acc: 0.9966 -
val_loss: 0.0875 - val_acc: 0.9826
Epoch 22/30
60000/60000 [==============================] - 8s 133us/step - loss: 0.0089 - acc: 0.9974 -
val_loss: 0.1040 - val_acc: 0.9791
Epoch 23/30
60000/60000 [==============================] - 8s 133us/step - loss: 0.0097 - acc: 0.9973 -
val_loss: 0.1075 - val_acc: 0.9810
Epoch 24/30
60000/60000 [==============================] - 8s 134us/step - loss: 0.0102 - acc: 0.9971 -
val_loss: 0.1052 - val_acc: 0.9796
Epoch 25/30
60000/60000 [==============================] - 8s 132us/step - loss: 0.0109 - acc: 0.9967 -
val_loss: 0.0963 - val_acc: 0.9781
Epoch 26/30
60000/60000 [==============================] - 8s 132us/step - loss: 0.0067 - acc: 0.9979 -
val_loss: 0.0902 - val_acc: 0.9831
Epoch 27/30
60000/60000 [==============================] - 8s 138us/step - loss: 0.0069 - acc: 0.9983 -
val_loss: 0.1002 - val_acc: 0.9819
Epoch 28/30
60000/60000 [==============================] - 8s 133us/step - loss: 0.0093 - acc: 0.9974 -
val_loss: 0.0922 - val_acc: 0.9809
Epoch 29/30
60000/60000 [==============================] - 8s 132us/step - loss: 0.0088 - acc: 0.9975 -
val_loss: 0.1053 - val_acc: 0.9803
Epoch 30/30
60000/60000 [==============================] - 8s 132us/step - loss: 0.0088 - acc: 0.9977 -
val_loss: 0.0946 - val_acc: 0.9826
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

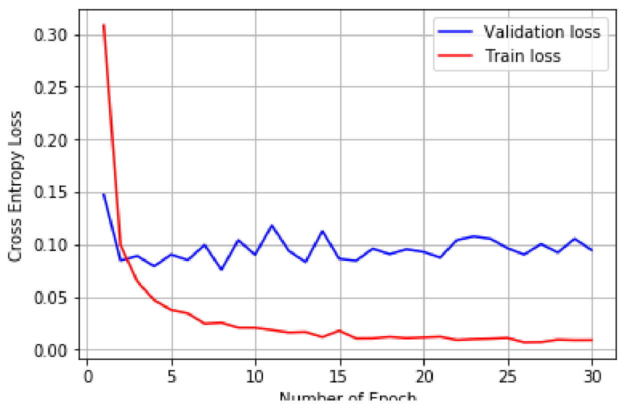
```
Test score: 0.09456962197794405
Test accuracy: 0.9826
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

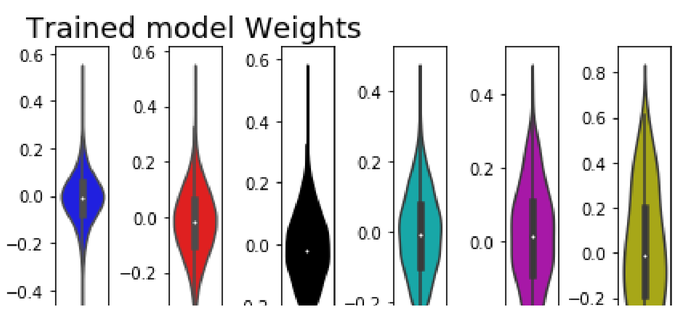
```
(784, 400)
(400,)
(400, 300)
(300,)
(300, 200)
(200,)
(200, 100)
(100,)
(100, 10)
(10,)
(10, 10)
(10,)
```

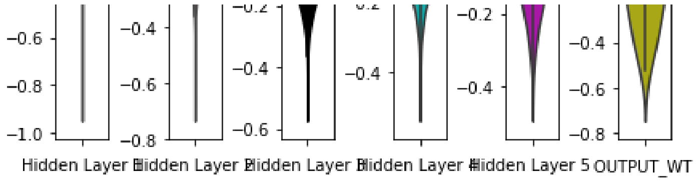
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
H3_WT = MODEL_WT[4].flatten().reshape(-1,1)
H4_WT = MODEL_WT[6].flatten().reshape(-1,1)
H5_WT = MODEL_WT[8].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[10].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4,axes5,axes6) = plt.subplots(nrows=1, ncols=6)
fig.tight_layout()

plt.subplot(1, 6, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 6, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 6, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 6, 4)
VIOLIN = SNS.violinplot(y=H4_WT, color='c')
plt.xlabel('Hidden Layer 4 ')

plt.subplot(1, 6, 5)
VIOLIN = SNS.violinplot(y=H5_WT, color='m')
plt.xlabel('Hidden Layer 5 ')

plt.subplot(1, 6, 6)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```





5 Hidden Layer MLP + Batch Normalization

Neuron in Hidden Layers = [400-300-200-100-50]

In [0]:

```
MODEL = Sequential()
MODEL.add(Dense(400,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(BatchNormalization())

MODEL.add(Dense(300,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())

MODEL.add(Dense(200,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())

MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())

MODEL.add(Dense(50,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())

MODEL.add(Dense(10,activation='softmax'))
print(MODEL.summary())

MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_132 (Dense)            (None, 400)               314000    
_________________________________________________________________
batch_normalization_54 (Batc (None, 400)               1600      
_________________________________________________________________
dense_133 (Dense)            (None, 300)               120300    
_________________________________________________________________
batch_normalization_55 (Batc (None, 300)               1200      
_________________________________________________________________
dense_134 (Dense)            (None, 200)               60200     
_________________________________________________________________
batch_normalization_56 (Batc (None, 200)               800       
_________________________________________________________________
dense_135 (Dense)            (None, 100)               20100     
_________________________________________________________________
batch_normalization_57 (Batc (None, 100)               400       
_________________________________________________________________
dense_136 (Dense)            (None, 50)                5050      
_________________________________________________________________
batch_normalization_58 (Batc (None, 50)                200       
_________________________________________________________________
dense_137 (Dense)            (None, 10)                510       
=================================================================
Total params: 524,360
Trainable params: 522,260
Non-trainable params: 2,100
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 16s 272us/step - loss: 0.2379 - acc: 0.9310 - val_l
oss: 0.1074 - val_acc: 0.9663
Epoch 2/30
```

```
60000/60000 [==============================] - 10s 172us/step - loss: 0.0908 - acc: 0.9718 - val_l
oss: 0.0894 - val_acc: 0.9726
Epoch 3/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0625 - acc: 0.9801 - val_l
oss: 0.0788 - val_acc: 0.9768
Epoch 4/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0517 - acc: 0.9839 - val_l
oss: 0.0892 - val_acc: 0.9722
Epoch 5/30
60000/60000 [==============================] - 10s 175us/step - loss: 0.0409 - acc: 0.9871 - val_l
oss: 0.0855 - val_acc: 0.9741
Epoch 6/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0368 - acc: 0.9875 - val_l
oss: 0.0854 - val_acc: 0.9740
Epoch 7/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0302 - acc: 0.9904 - val_l
oss: 0.0769 - val_acc: 0.9780
Epoch 8/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0303 - acc: 0.9896 - val_l
oss: 0.0887 - val_acc: 0.9750
Epoch 9/30
60000/60000 [==============================] - 11s 175us/step - loss: 0.0248 - acc: 0.9920 - val_l
oss: 0.0821 - val_acc: 0.9769
Epoch 10/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0228 - acc: 0.9923 - val_l
oss: 0.0774 - val_acc: 0.9795
Epoch 11/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0215 - acc: 0.9930 - val_l
oss: 0.0836 - val_acc: 0.9787
Epoch 12/30
60000/60000 [==============================] - 10s 175us/step - loss: 0.0200 - acc: 0.9936 - val_l
oss: 0.0793 - val_acc: 0.9777
Epoch 13/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0186 - acc: 0.9940 - val_l
oss: 0.0907 - val_acc: 0.9775
Epoch 14/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0175 - acc: 0.9942 - val_l
oss: 0.0908 - val_acc: 0.9785
Epoch 15/30
60000/60000 [==============================] - 11s 175us/step - loss: 0.0177 - acc: 0.9941 - val_l
oss: 0.0789 - val_acc: 0.9791
Epoch 16/30
60000/60000 [==============================] - 10s 175us/step - loss: 0.0146 - acc: 0.9954 - val_l
oss: 0.0742 - val_acc: 0.9800
Epoch 17/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0140 - acc: 0.9952 - val_l
oss: 0.0723 - val_acc: 0.9796
Epoch 18/30
60000/60000 [==============================] - 10s 175us/step - loss: 0.0135 - acc: 0.9957 - val_l
oss: 0.0841 - val_acc: 0.9787
Epoch 19/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0145 - acc: 0.9953 - val_l
oss: 0.0811 - val_acc: 0.9799
Epoch 20/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0133 - acc: 0.9956 - val_l
oss: 0.0743 - val_acc: 0.9819
Epoch 21/30
60000/60000 [==============================] - 11s 179us/step - loss: 0.0112 - acc: 0.9964 - val_l
oss: 0.0859 - val_acc: 0.9795
Epoch 22/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0107 - acc: 0.9964 - val_l
oss: 0.0799 - val_acc: 0.9831
Epoch 23/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0120 - acc: 0.9960 - val_l
oss: 0.0720 - val_acc: 0.9817
Epoch 24/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0136 - acc: 0.9957 - val_l
oss: 0.0717 - val_acc: 0.9816
Epoch 25/30
60000/60000 [==============================] - 11s 175us/step - loss: 0.0090 - acc: 0.9969 - val_l
oss: 0.0644 - val_acc: 0.9818
Epoch 26/30
60000/60000 [==============================] - 11s 180us/step - loss: 0.0081 - acc: 0.9974 - val_l
oss: 0.0821 - val_acc: 0.9802
Epoch 27/30
60000/60000 [==============================] - 11s 178us/step - loss: 0.0107 - acc: 0.9966 - val_l
oss: 0.0808 - val_acc: 0.9800
```

```
Epoch 28/30
60000/60000 [==============================] - 11s 175us/step - loss: 0.0083 - acc: 0.9974 - val_l
oss: 0.0629 - val_acc: 0.9829
Epoch 29/30
60000/60000 [==============================] - 11s 179us/step - loss: 0.0085 - acc: 0.9972 - val_l
oss: 0.0775 - val_acc: 0.9811
Epoch 30/30
60000/60000 [==============================] - 11s 176us/step - loss: 0.0088 - acc: 0.9969 - val_l
oss: 0.0698 - val_acc: 0.9829
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

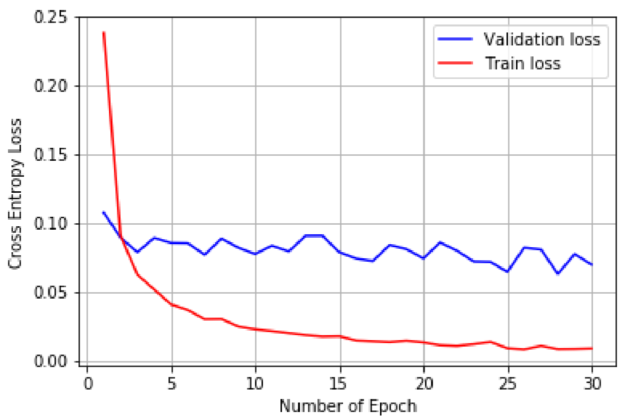
```
Test score: 0.06982514466176945
Test accuracy: 0.9829
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 400)
(400,)
(400,)
(400,)
(400,)
(400,)
(400, 300)
(300,)
(300,)
(300,)
(300,)
(300,)
(300, 200)
```

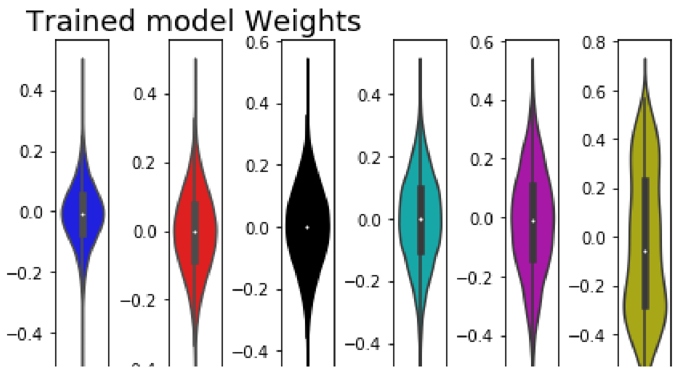
```
(200,)
(200,)
(200,)
(200,)
(200,)
(200, 100)
(100,)
(100,)
(100,)
(100,)
(100,)
(100, 50)
(50,)
(50,)
(50,)
(50,)
(50,)
(50, 10)
(10,)
```

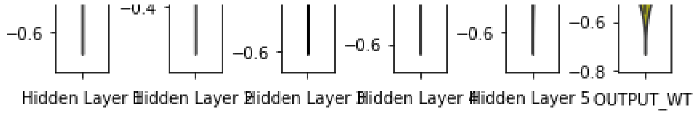
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
H4_WT = MODEL_WT[18].flatten().reshape(-1,1)
H5_WT = MODEL_WT[24].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[30].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4,axes5,axes6) = plt.subplots(nrows=1, ncols=6)
fig.tight_layout()

plt.subplot(1, 6, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 6, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 6, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 6, 4)
VIOLIN = SNS.violinplot(y=H4_WT, color='c')
plt.xlabel('Hidden Layer 4 ')

plt.subplot(1, 6, 5)
VIOLIN = SNS.violinplot(y=H5_WT, color='m')
plt.xlabel('Hidden Layer 5 ')

plt.subplot(1, 6, 6)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```





5 Hidden Layer MLP having no Dropout and Batch Normalization

Neuron in Hidden Layers = [400-300-200-100-50]

In [0]:

```
MODEL = Sequential()
MODEL.add(Dense(400,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=he_normal(
seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))

MODEL.add(Dense(300,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))

MODEL.add(Dense(200,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))

MODEL.add(Dense(100,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))

MODEL.add(Dense(50,activation='relu',kernel_initializer=he_normal(seed=None)))
MODEL.add(BatchNormalization())
MODEL.add(Dropout(0.5))

MODEL.add(Dense(10,activation='softmax'))
print(MODEL.summary())

MODEL.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
FINAL_MODEL = MODEL.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,
Y_TEST))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_138 (Dense)            (None, 400)               314000    
_________________________________________________________________
batch_normalization_59 (Batc (None, 400)               1600      
_________________________________________________________________
dropout_37 (Dropout)         (None, 400)               0         
_________________________________________________________________
dense_139 (Dense)            (None, 300)               120300    
_________________________________________________________________
batch_normalization_60 (Batc (None, 300)               1200      
_________________________________________________________________
dropout_38 (Dropout)         (None, 300)               0         
_________________________________________________________________
dense_140 (Dense)            (None, 200)               60200     
_________________________________________________________________
batch_normalization_61 (Batc (None, 200)               800       
_________________________________________________________________
dropout_39 (Dropout)         (None, 200)               0         
_________________________________________________________________
dense_141 (Dense)            (None, 100)               20100     
_________________________________________________________________
batch_normalization_62 (Batc (None, 100)               400       
_________________________________________________________________
dropout_40 (Dropout)         (None, 100)               0         
_________________________________________________________________
dense_142 (Dense)            (None, 50)                5050      
_________________________________________________________________
batch_normalization_63 (Batc (None, 50)                200       
```

```
_________________________________________________________________
dropout_41 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_143 (Dense)            (None, 10)                510
=================================================================
Total params: 524,360
Trainable params: 522,260
Non-trainable params: 2,100
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 18s 303us/step - loss: 1.3196 - acc: 0.5737 - val_l
oss: 0.2779 - val_acc: 0.9206
Epoch 2/30
60000/60000 [==============================] - 11s 190us/step - loss: 0.4706 - acc: 0.8704 - val_l
oss: 0.1867 - val_acc: 0.9509
Epoch 3/30
60000/60000 [==============================] - 11s 189us/step - loss: 0.3342 - acc: 0.9152 - val_l
oss: 0.1534 - val_acc: 0.9581
Epoch 4/30
60000/60000 [==============================] - 11s 191us/step - loss: 0.2759 - acc: 0.9301 - val_l
oss: 0.1352 - val_acc: 0.9642
Epoch 5/30
60000/60000 [==============================] - 12s 194us/step - loss: 0.2423 - acc: 0.9394 - val_l
oss: 0.1232 - val_acc: 0.9686
Epoch 6/30
60000/60000 [==============================] - 12s 194us/step - loss: 0.2208 - acc: 0.9447 - val_l
oss: 0.1068 - val_acc: 0.9731
Epoch 7/30
60000/60000 [==============================] - 12s 195us/step - loss: 0.2052 - acc: 0.9490 - val_l
oss: 0.1091 - val_acc: 0.9710
Epoch 8/30
60000/60000 [==============================] - 12s 194us/step - loss: 0.1865 - acc: 0.9539 - val_l
oss: 0.0996 - val_acc: 0.9737
Epoch 9/30
60000/60000 [==============================] - 12s 192us/step - loss: 0.1737 - acc: 0.9569 - val_l
oss: 0.0977 - val_acc: 0.9747
Epoch 10/30
60000/60000 [==============================] - 14s 229us/step - loss: 0.1695 - acc: 0.9579 - val_l
oss: 0.0991 - val_acc: 0.9746
Epoch 11/30
60000/60000 [==============================] - 12s 200us/step - loss: 0.1549 - acc: 0.9621 - val_l
oss: 0.0861 - val_acc: 0.9767
Epoch 12/30
60000/60000 [==============================] - 12s 199us/step - loss: 0.1484 - acc: 0.9638 - val_l
oss: 0.0781 - val_acc: 0.9796
Epoch 13/30
60000/60000 [==============================] - 12s 201us/step - loss: 0.1392 - acc: 0.9648 - val_l
oss: 0.0815 - val_acc: 0.9784
Epoch 14/30
60000/60000 [==============================] - 12s 200us/step - loss: 0.1374 - acc: 0.9658 - val_l
oss: 0.0792 - val_acc: 0.9791
Epoch 15/30
60000/60000 [==============================] - 12s 200us/step - loss: 0.1339 - acc: 0.9673 - val_l
oss: 0.0855 - val_acc: 0.9778
Epoch 16/30
60000/60000 [==============================] - 12s 203us/step - loss: 0.1279 - acc: 0.9679 - val_l
oss: 0.0810 - val_acc: 0.9810
Epoch 17/30
60000/60000 [==============================] - 12s 207us/step - loss: 0.1267 - acc: 0.9682 - val_l
oss: 0.0764 - val_acc: 0.9814
Epoch 18/30
60000/60000 [==============================] - 12s 202us/step - loss: 0.1216 - acc: 0.9701 - val_l
oss: 0.0858 - val_acc: 0.9787
Epoch 19/30
60000/60000 [==============================] - 12s 201us/step - loss: 0.1136 - acc: 0.9716 - val_l
oss: 0.0810 - val_acc: 0.9793
Epoch 20/30
60000/60000 [==============================] - 12s 202us/step - loss: 0.1147 - acc: 0.9716 - val_l
oss: 0.0733 - val_acc: 0.9815
Epoch 21/30
60000/60000 [==============================] - 12s 207us/step - loss: 0.1075 - acc: 0.9741 - val_l
oss: 0.0742 - val_acc: 0.9824
Epoch 22/30
60000/60000 [==============================] - 12s 205us/step - loss: 0.1059 - acc: 0.9737 - val_l
oss: 0.0743 - val_acc: 0.9829
Epoch 23/30
```

```
60000/60000 [==============================] - 12s 198us/step - loss: 0.1054 - acc: 0.9734 - val_l
oss: 0.0767 - val_acc: 0.9821
Epoch 24/30
60000/60000 [==============================] - 12s 203us/step - loss: 0.0991 - acc: 0.9747 - val_l
oss: 0.0740 - val_acc: 0.9835
Epoch 25/30
60000/60000 [==============================] - 12s 205us/step - loss: 0.0978 - acc: 0.9753 - val_l
oss: 0.0675 - val_acc: 0.9838
Epoch 26/30
60000/60000 [==============================] - 12s 202us/step - loss: 0.0975 - acc: 0.9764 - val_l
oss: 0.0737 - val_acc: 0.9816
Epoch 27/30
60000/60000 [==============================] - 13s 214us/step - loss: 0.0920 - acc: 0.9775 - val_l
oss: 0.0702 - val_acc: 0.9833
Epoch 28/30
60000/60000 [==============================] - 12s 202us/step - loss: 0.0928 - acc: 0.9767 - val_l
oss: 0.0623 - val_acc: 0.9854
Epoch 29/30
60000/60000 [==============================] - 12s 202us/step - loss: 0.0915 - acc: 0.9773 - val_l
oss: 0.0733 - val_acc: 0.9825
Epoch 30/30
60000/60000 [==============================] - 12s 200us/step - loss: 0.0889 - acc: 0.9778 - val_l
oss: 0.0733 - val_acc: 0.9838
```

In [0]:

```
score = MODEL.evaluate(X_TEST, Y_TEST, verbose=0)
TEST_ACCURACY.append(score[1])
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

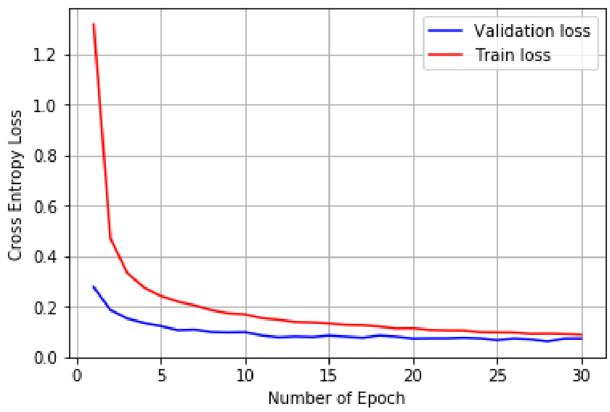
```
Test score: 0.07330910813587252
Test accuracy: 0.9838
```

In [0]:

```
TRAIN_LOSS = FINAL_MODEL.history['loss']
VAL_LOSS = FINAL_MODEL.history['val_loss']
X = list(range(1,31))
plt.plot(X,VAL_LOSS,'b',label="Validation loss")
plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
plt.legend()
plt.grid()
plt.xlabel("Number of Epoch")
plt.ylabel('Cross Entropy Loss')
```

Out[0]:

```
Text(0, 0.5, 'Cross Entropy Loss')
```



In [0]:

```
MODEL_WT = MODEL.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

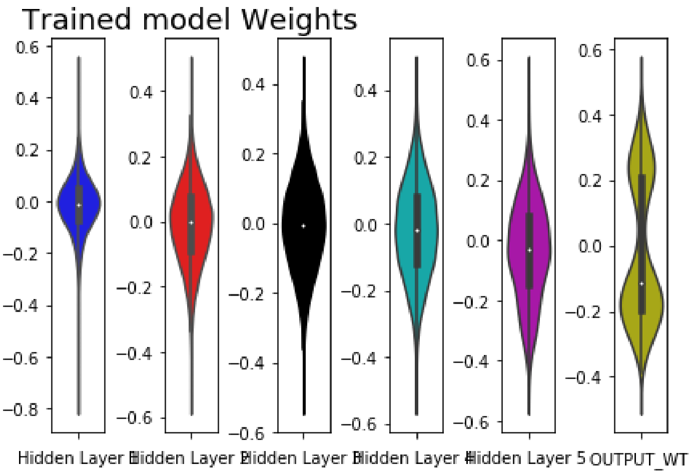
```
(784, 400)
(400,)
(400,)
(400,)
(400,)
(400,)
(400, 300)
(300,)
(300,)
(300,)
(300,)
(300,)
(300, 200)
(200,)
(200,)
(200,)
(200,)
(200,)
(200, 100)
(100,)
(100,)
(100,)
(100,)
(100,)
(100, 50)
(50,)
(50,)
(50,)
(50,)
(50,)
(50, 10)
(10,)
```

In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
H4_WT = MODEL_WT[18].flatten().reshape(-1,1)
H5_WT = MODEL_WT[24].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[30].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4,axes5,axes6) = plt.subplots(nrows=1, ncols=6)
fig.tight_layout()

plt.subplot(1, 6, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 6, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 6, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 6, 4)
VIOLIN = SNS.violinplot(y=H4_WT, color='c')
plt.xlabel('Hidden Layer 4 ')

plt.subplot(1, 6, 5)
VIOLIN = SNS.violinplot(y=H5_WT, color='m')
plt.xlabel('Hidden Layer 5 ')

plt.subplot(1, 6, 6)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



In [0]:

```python
from prettytable import PrettyTable
```

In [2]:

```python
X=PrettyTable()
print(" "*40+"CONCLUSION")
print("="*100)
X.field_names = ["Model","Number Of Hidden Layers","Neurons in Layer","Test Loss"]
X.add_row(["MLP+RELU+ADAM","2","[200-100]",0.095])
X.add_row(["MLP+RELU+ADAM+Batch Normalization ","2","[200-100]",0.0961])
X.add_row(["MLP+RELU+ADAM+Batch Normalization+Dropout","2","[200-100]",0.065])

X.add_row(["MLP+RELU+ADAM","3","[200-100-50]",0.0931])
X.add_row(["MLP+RELU+ADAM+Batch Normalization ","3","[200-100-50]",0.854])
X.add_row(["MLP+RELU+ADAM+Batch Normalization+Dropout","3","[200-100-50]",0.0737])

X.add_row(["MLP+RELU+ADAM","5","[400-300-200-100-50]",0.0945])
X.add_row(["MLP+RELU+ADAM+Batch Normalization ","5","[400-300-200-100-50]",0.0698])
X.add_row(["MLP+RELU+ADAM+Batch Normalization+Dropout","5","[400-300-200-100-50]",0.0733])
print(X)
```

```
                                        CONCLUSION
====================================================================================================

+-------------------------------------------+-------------------------+----------------------+-----
--+
|                   Model                   | Number Of Hidden Layers |   Neurons in Layer   | Test
Loss |
+-------------------------------------------+-------------------------+----------------------+-----
--+
|               MLP+RELU+ADAM               |            2            |      [200-100]       |   0.
5   |
|     MLP+RELU+ADAM+Batch Normalization     |            2            |      [200-100]       |   0.
961   |
| MLP+RELU+ADAM+Batch Normalization+Dropout |            2            |      [200-100]       |   0.
65    |
|               MLP+RELU+ADAM               |            3            |     [200-100-50]     |   0.
31   |
|     MLP+RELU+ADAM+Batch Normalization     |            3            |     [200-100-50]     |   0.
54    |
| MLP+RELU+ADAM+Batch Normalization+Dropout |            3            |     [200-100-50]     |   0.
737   |
|               MLP+RELU+ADAM               |            5            | [400-300-200-100-50] |   0.
45   |
|     MLP+RELU+ADAM+Batch Normalization     |            5            | [400-300-200-100-50] |   0.
698   |
| MLP+RELU+ADAM+Batch Normalization+Dropout |            5            | [400-300-200-100-50] |   0.
0733   |
+-------------------------------------------+-------------------------+----------------------+-----
--+
```

Updated ....Part:

Model:1

Activation Function: sigmoid

Optimizer: SGD

Wt initializer : Random uniform

MLP Architecture: [240-120-60]

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.initializers import RandomNormal
```

In [0]:

```
M= Sequential()
M.add(Dense(240,activation='sigmoid',input_shape=(X_TRAIN.shape[1],),kernel_initializer=RandomNorma
l(mean=0,stddev=1,seed=None)))
M.add(Dense(120,activation='sigmoid',kernel_initializer=RandomNormal(mean=0,stddev=1,seed=None)))
M.add(Dense(60,activation='sigmoid',kernel_initializer=RandomNormal(mean=0,stddev=1,seed=None)))
M.add(Dense(10,activation='softmax'))
```

In [0]:

```
def NN(MODEL3,OPTIMIZER):
  TEST_ACCURACY=[]
  MODEL3.compile(optimizer=OPTIMIZER,loss='categorical_crossentropy',metrics=['accuracy'])
  History = MODEL3.fit(X_TRAIN,Y_TRAIN,batch_size=128,epochs=30,verbose=1,validation_data=(X_TEST,Y
_TEST))
  score = MODEL3.evaluate(X_TEST, Y_TEST, verbose=0)
  TEST_ACCURACY.append(score[1])
  print('Test score:', score[0])
  print('Test accuracy:', score[1])
  return History

def PLOT(History):
  TRAIN_LOSS = History.history['loss']
  VAL_LOSS = History.history['val_loss']
  X = list(range(1,31))
  plt.plot(X,VAL_LOSS,'b',label="Validation loss")
  plt.plot(X,TRAIN_LOSS,'r',label='Train loss')
  plt.legend()
  plt.grid()
  plt.xlabel("Number of Epoch")
  plt.ylabel('Cross Entropy Loss')
```

In [0]:

```
H=NN(M,'SGD')
```

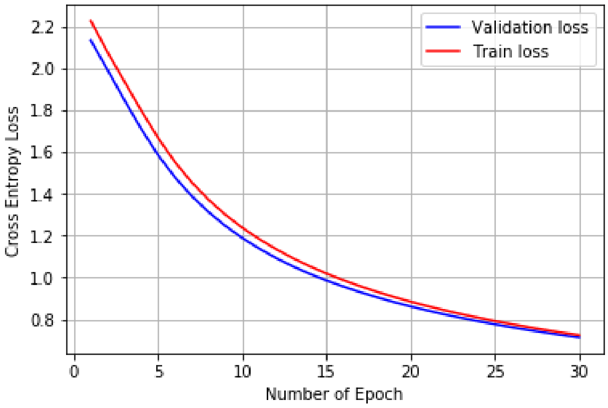
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 10s 158us/step - loss: 2.2261 - acc: 0.1883 - val_l
oss: 2.1329 - val_acc: 0.2452
Epoch 2/30
60000/60000 [==============================] - 4s 65us/step - loss: 2.0760 - acc: 0.2892 -
val_loss: 1.9912 - val_acc: 0.3457
Epoch 3/30
60000/60000 [==============================] - 4s 65us/step - loss: 1.9363 - acc: 0.3836 -
val_loss: 1.8475 - val_acc: 0.4262
Epoch 4/30
60000/60000 [==============================] - 4s 65us/step - loss: 1.7967 - acc: 0.4508 -
```

```
val_loss: 1.7086 - val_acc: 0.4794
Epoch 5/30
60000/60000 [==============================] - 4s 66us/step - loss: 1.6662 - acc: 0.5005 -
val_loss: 1.5844 - val_acc: 0.5226
Epoch 6/30
60000/60000 [==============================] - 4s 66us/step - loss: 1.5518 - acc: 0.5360 -
val_loss: 1.4787 - val_acc: 0.5560
Epoch 7/30
60000/60000 [==============================] - 4s 66us/step - loss: 1.4542 - acc: 0.5670 -
val_loss: 1.3885 - val_acc: 0.5859
Epoch 8/30
60000/60000 [==============================] - 4s 67us/step - loss: 1.3709 - acc: 0.5903 -
val_loss: 1.3121 - val_acc: 0.6098
Epoch 9/30
60000/60000 [==============================] - 4s 66us/step - loss: 1.2995 - acc: 0.6105 -
val_loss: 1.2458 - val_acc: 0.6278
Epoch 10/30
60000/60000 [==============================] - 4s 64us/step - loss: 1.2376 - acc: 0.6271 -
val_loss: 1.1884 - val_acc: 0.6435
Epoch 11/30
60000/60000 [==============================] - 4s 65us/step - loss: 1.1833 - acc: 0.6427 -
val_loss: 1.1377 - val_acc: 0.6583
Epoch 12/30
60000/60000 [==============================] - 4s 62us/step - loss: 1.1352 - acc: 0.6565 -
val_loss: 1.0929 - val_acc: 0.6698
Epoch 13/30
60000/60000 [==============================] - 4s 62us/step - loss: 1.0924 - acc: 0.6674 -
val_loss: 1.0534 - val_acc: 0.6813
Epoch 14/30
60000/60000 [==============================] - 4s 67us/step - loss: 1.0538 - acc: 0.6774 -
val_loss: 1.0177 - val_acc: 0.6911
Epoch 15/30
60000/60000 [==============================] - 4s 67us/step - loss: 1.0189 - acc: 0.6874 -
val_loss: 0.9854 - val_acc: 0.6993
Epoch 16/30
60000/60000 [==============================] - 4s 66us/step - loss: 0.9871 - acc: 0.6974 -
val_loss: 0.9565 - val_acc: 0.7059
Epoch 17/30
60000/60000 [==============================] - 4s 65us/step - loss: 0.9582 - acc: 0.7051 -
val_loss: 0.9298 - val_acc: 0.7113
Epoch 18/30
60000/60000 [==============================] - 4s 66us/step - loss: 0.9315 - acc: 0.7128 -
val_loss: 0.9053 - val_acc: 0.7204
Epoch 19/30
60000/60000 [==============================] - 4s 65us/step - loss: 0.9070 - acc: 0.7196 -
val_loss: 0.8823 - val_acc: 0.7261
Epoch 20/30
60000/60000 [==============================] - 4s 65us/step - loss: 0.8842 - acc: 0.7272 -
val_loss: 0.8616 - val_acc: 0.7301
Epoch 21/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.8632 - acc: 0.7324 -
val_loss: 0.8420 - val_acc: 0.7361
Epoch 22/30
60000/60000 [==============================] - 4s 63us/step - loss: 0.8435 - acc: 0.7376 -
val_loss: 0.8237 - val_acc: 0.7431
Epoch 23/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.8251 - acc: 0.7442 -
val_loss: 0.8070 - val_acc: 0.7479
Epoch 24/30
60000/60000 [==============================] - 4s 63us/step - loss: 0.8080 - acc: 0.7491 -
val_loss: 0.7909 - val_acc: 0.7514
Epoch 25/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.7919 - acc: 0.7543 -
val_loss: 0.7762 - val_acc: 0.7560
Epoch 26/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.7768 - acc: 0.7591 -
val_loss: 0.7617 - val_acc: 0.7609
Epoch 27/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.7625 - acc: 0.7627 -
val_loss: 0.7487 - val_acc: 0.7648
Epoch 28/30
60000/60000 [==============================] - 4s 62us/step - loss: 0.7490 - acc: 0.7674 -
val_loss: 0.7364 - val_acc: 0.7690
Epoch 29/30
60000/60000 [==============================] - 4s 63us/step - loss: 0.7364 - acc: 0.7705 -
val_loss: 0.7244 - val_acc: 0.7726
Epoch 30/30
```

```
60000/60000 [==============================] - 4s 63us/step - loss: 0.7242 - acc: 0.7740 -
val_loss: 0.7130 - val_acc: 0.7754
Test score: 0.7130339228630066
Test accuracy: 0.7754
```

In [0]:

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

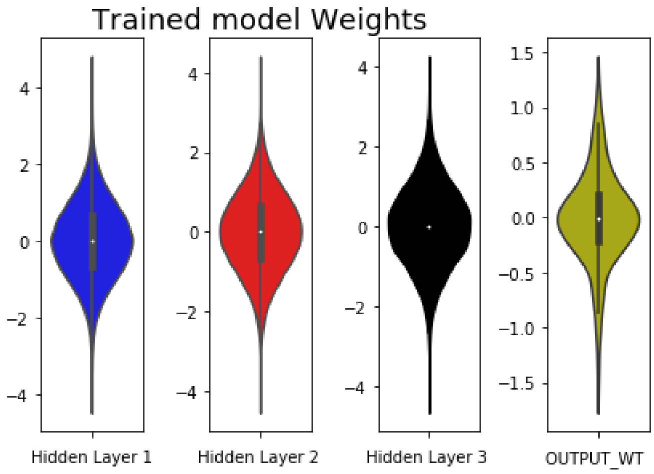
```
(784, 240)
(240,)
(240, 120)
(120,)
(120, 60)
(60,)
(60, 10)
(10,)
```

In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
H3_WT = MODEL_WT[4].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[6].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')


plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



MODEL:2

Activation Function: Tanh

Optimizer: Adagrad

Wt initializer : Random Normal

MLP Architecture: [340-150-20]

In [0]:

```
from keras.initializers import RandomUniform
M= Sequential()
M.add(Dense(340,activation='tanh',input_shape=(X_TRAIN.shape[1],),kernel_initializer=RandomUniform(
minval=-0.05, maxval=0.05, seed=None)))
M.add(Dense(150,activation='tanh',kernel_initializer=RandomUniform(minval=-0.05, maxval=0.05, seed=
None)))
M.add(Dense(20,activation='tanh',kernel_initializer=RandomUniform(minval=-0.05, maxval=0.05, seed=N
one)))
M.add(Dense(10,activation='softmax'))
```

In [0]:

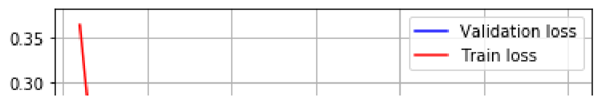
```
H=NN(M,'Adagrad')
```

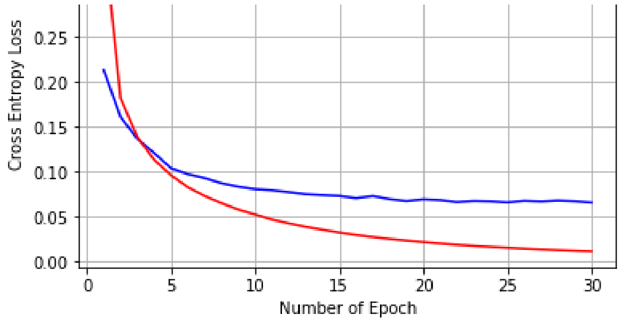
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 11s 181us/step - loss: 0.3647 - acc: 0.9027 - val_l
oss: 0.2133 - val_acc: 0.9410
Epoch 2/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.1820 - acc: 0.9498 -
val_loss: 0.1610 - val_acc: 0.9539
Epoch 3/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.1387 - acc: 0.9612 -
val_loss: 0.1367 - val_acc: 0.9595
Epoch 4/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.1133 - acc: 0.9691 -
val_loss: 0.1208 - val_acc: 0.9648
Epoch 5/30
60000/60000 [==============================] - 5s 85us/step - loss: 0.0961 - acc: 0.9742 -
val_loss: 0.1039 - val_acc: 0.9684
Epoch 6/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0828 - acc: 0.9778 -
val_loss: 0.0974 - val_acc: 0.9711
Epoch 7/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0731 - acc: 0.9801 -
val_loss: 0.0930 - val_acc: 0.9727
Epoch 8/30
60000/60000 [==============================] - 5s 89us/step - loss: 0.0651 - acc: 0.9830 -
val_loss: 0.0872 - val_acc: 0.9743
Epoch 9/30
```

```
60000/60000 [==============================] - 5s 89us/step - loss: 0.0579 - acc: 0.9848 -
val_loss: 0.0834 - val_acc: 0.9749
Epoch 10/30
60000/60000 [==============================] - 5s 89us/step - loss: 0.0524 - acc: 0.9863 -
val_loss: 0.0807 - val_acc: 0.9766
Epoch 11/30
60000/60000 [==============================] - 5s 89us/step - loss: 0.0468 - acc: 0.9882 -
val_loss: 0.0795 - val_acc: 0.9765
Epoch 12/30
60000/60000 [==============================] - 5s 92us/step - loss: 0.0425 - acc: 0.9894 -
val_loss: 0.0773 - val_acc: 0.9772
Epoch 13/30
60000/60000 [==============================] - 6s 92us/step - loss: 0.0387 - acc: 0.9907 -
val_loss: 0.0751 - val_acc: 0.9774
Epoch 14/30
60000/60000 [==============================] - 5s 91us/step - loss: 0.0355 - acc: 0.9918 -
val_loss: 0.0740 - val_acc: 0.9782
Epoch 15/30
60000/60000 [==============================] - 5s 88us/step - loss: 0.0323 - acc: 0.9925 -
val_loss: 0.0732 - val_acc: 0.9772
Epoch 16/30
60000/60000 [==============================] - 5s 89us/step - loss: 0.0296 - acc: 0.9936 -
val_loss: 0.0706 - val_acc: 0.9790
Epoch 17/30
60000/60000 [==============================] - 5s 90us/step - loss: 0.0273 - acc: 0.9944 -
val_loss: 0.0730 - val_acc: 0.9780
Epoch 18/30
60000/60000 [==============================] - 5s 89us/step - loss: 0.0252 - acc: 0.9950 -
val_loss: 0.0696 - val_acc: 0.9797
Epoch 19/30
60000/60000 [==============================] - 5s 88us/step - loss: 0.0233 - acc: 0.9957 -
val_loss: 0.0674 - val_acc: 0.9803
Epoch 20/30
60000/60000 [==============================] - 5s 90us/step - loss: 0.0217 - acc: 0.9959 -
val_loss: 0.0693 - val_acc: 0.9793
Epoch 21/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0201 - acc: 0.9965 -
val_loss: 0.0685 - val_acc: 0.9796
Epoch 22/30
60000/60000 [==============================] - 5s 88us/step - loss: 0.0187 - acc: 0.9970 -
val_loss: 0.0663 - val_acc: 0.9805
Epoch 23/30
60000/60000 [==============================] - 5s 89us/step - loss: 0.0174 - acc: 0.9972 -
val_loss: 0.0675 - val_acc: 0.9801
Epoch 24/30
60000/60000 [==============================] - 5s 89us/step - loss: 0.0163 - acc: 0.9975 -
val_loss: 0.0671 - val_acc: 0.9804
Epoch 25/30
60000/60000 [==============================] - 5s 86us/step - loss: 0.0153 - acc: 0.9977 -
val_loss: 0.0659 - val_acc: 0.9806
Epoch 26/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0142 - acc: 0.9981 -
val_loss: 0.0677 - val_acc: 0.9799
Epoch 27/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0134 - acc: 0.9981 -
val_loss: 0.0669 - val_acc: 0.9799
Epoch 28/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0126 - acc: 0.9983 -
val_loss: 0.0680 - val_acc: 0.9801
Epoch 29/30
60000/60000 [==============================] - 5s 88us/step - loss: 0.0119 - acc: 0.9986 -
val_loss: 0.0672 - val_acc: 0.9802
Epoch 30/30
60000/60000 [==============================] - 5s 87us/step - loss: 0.0112 - acc: 0.9987 -
val_loss: 0.0657 - val_acc: 0.9811
Test score: 0.06574692367911339
Test accuracy: 0.9811
```

In [0]:

```
PLOT(H)
```





In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

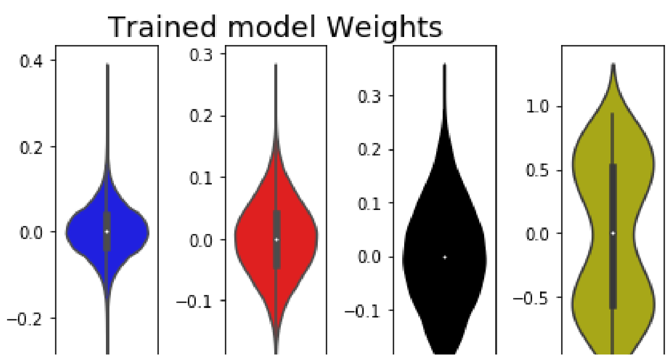
```
(784, 340)
(340,)
(340, 150)
(150,)
(150, 20)
(20,)
(20, 10)
(10,)
```

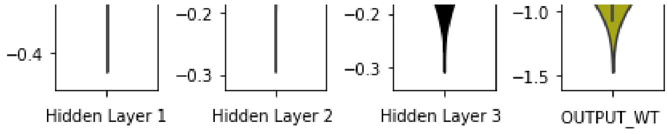
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
H3_WT = MODEL_WT[4].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[6].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')


plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```





MODEL:3

Activation Function: hard_sigmoid

Optimizer: Adadelta

Wt initializer : glorot_normal

MLP Architecture: [420-300-200-120-20]

In [0]:

```
from keras.initializers import glorot_normal
M= Sequential()
M.add(Dense(420,activation='hard_sigmoid',input_shape=(X_TRAIN.shape[1],),kernel_initializer=glorot
_normal(seed=None)))
M.add(Dense(300,activation='hard_sigmoid',kernel_initializer=glorot_normal(seed=None)))
M.add(Dense(200,activation='hard_sigmoid',kernel_initializer=glorot_normal(seed=None)))
M.add(Dense(120,activation='hard_sigmoid',kernel_initializer=glorot_normal(seed=None)))
M.add(Dense(20,activation='hard_sigmoid',kernel_initializer=glorot_normal(seed=None)))

M.add(Dense(10,activation='softmax'))
```

In [0]:

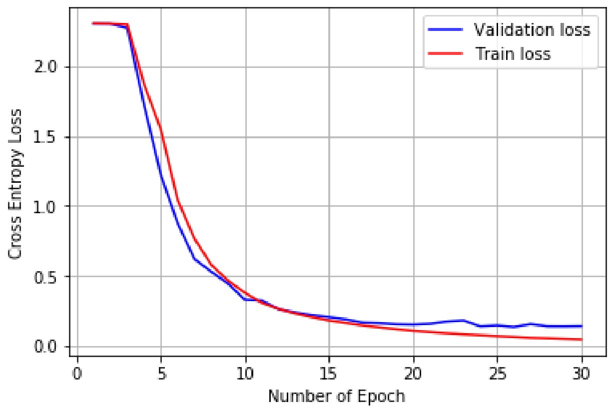
```
H=NN(M,'Adadelta')
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 16s 267us/step - loss: 2.3042 - acc: 0.1098 - val_l
oss: 2.3016 - val_acc: 0.1032
Epoch 2/30
60000/60000 [==============================] - 10s 169us/step - loss: 2.3025 - acc: 0.1095 - val_l
oss: 2.3015 - val_acc: 0.1010
Epoch 3/30
60000/60000 [==============================] - 10s 171us/step - loss: 2.2970 - acc: 0.1226 - val_l
oss: 2.2717 - val_acc: 0.1884
Epoch 4/30
60000/60000 [==============================] - 10s 170us/step - loss: 1.8693 - acc: 0.2952 - val_l
oss: 1.7211 - val_acc: 0.3263
Epoch 5/30
60000/60000 [==============================] - 10s 171us/step - loss: 1.5454 - acc: 0.4324 - val_l
oss: 1.2215 - val_acc: 0.5707
Epoch 6/30
60000/60000 [==============================] - 10s 171us/step - loss: 1.0441 - acc: 0.6608 - val_l
oss: 0.8775 - val_acc: 0.7375
Epoch 7/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.7662 - acc: 0.7772 - val_l
oss: 0.6220 - val_acc: 0.8362
Epoch 8/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.5808 - acc: 0.8423 - val_l
oss: 0.5316 - val_acc: 0.8587
Epoch 9/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.4675 - acc: 0.8781 - val_l
oss: 0.4466 - val_acc: 0.8839
Epoch 10/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.3819 - acc: 0.9019 - val_l
oss: 0.3312 - val_acc: 0.9181
Epoch 11/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.3093 - acc: 0.9194 - val_l
oss: 0.3259 - val_acc: 0.9118
Epoch 12/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.2661 - acc: 0.9305 - val_l
oss: 0.2642 - val_acc: 0.9323
Epoch 13/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.2331 - acc: 0.9384 - val_l
oss: 0.2401 - val_acc: 0.9387
```

```
Epoch 14/30
60000/60000 [==============================] - 11s 176us/step - loss: 0.2068 - acc: 0.9457 - val_l
oss: 0.2210 - val_acc: 0.9425
Epoch 15/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.1832 - acc: 0.9520 - val_l
oss: 0.2073 - val_acc: 0.9479
Epoch 16/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.1653 - acc: 0.9562 - val_l
oss: 0.1911 - val_acc: 0.9516
Epoch 17/30
60000/60000 [==============================] - 10s 168us/step - loss: 0.1479 - acc: 0.9609 - val_l
oss: 0.1681 - val_acc: 0.9564
Epoch 18/30
60000/60000 [==============================] - 10s 169us/step - loss: 0.1340 - acc: 0.9643 - val_l
oss: 0.1644 - val_acc: 0.9592
Epoch 19/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.1211 - acc: 0.9681 - val_l
oss: 0.1563 - val_acc: 0.9606
Epoch 20/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.1094 - acc: 0.9716 - val_l
oss: 0.1541 - val_acc: 0.9603
Epoch 21/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.1006 - acc: 0.9730 - val_l
oss: 0.1595 - val_acc: 0.9581
Epoch 22/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0919 - acc: 0.9762 - val_l
oss: 0.1758 - val_acc: 0.9523
Epoch 23/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0845 - acc: 0.9779 - val_l
oss: 0.1833 - val_acc: 0.9511
Epoch 24/30
60000/60000 [==============================] - 10s 169us/step - loss: 0.0779 - acc: 0.9797 - val_l
oss: 0.1416 - val_acc: 0.9631
Epoch 25/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.0711 - acc: 0.9814 - val_l
oss: 0.1481 - val_acc: 0.9618
Epoch 26/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.0656 - acc: 0.9833 - val_l
oss: 0.1368 - val_acc: 0.9651
Epoch 27/30
60000/60000 [==============================] - 10s 168us/step - loss: 0.0592 - acc: 0.9852 - val_l
oss: 0.1578 - val_acc: 0.9587
Epoch 28/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.0556 - acc: 0.9861 - val_l
oss: 0.1420 - val_acc: 0.9653
Epoch 29/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.0513 - acc: 0.9872 - val_l
oss: 0.1414 - val_acc: 0.9639
Epoch 30/30
60000/60000 [==============================] - 10s 169us/step - loss: 0.0473 - acc: 0.9885 - val_l
oss: 0.1428 - val_acc: 0.9621
Test score: 0.14281796935126184
Test accuracy: 0.9621
```

In [0]:

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

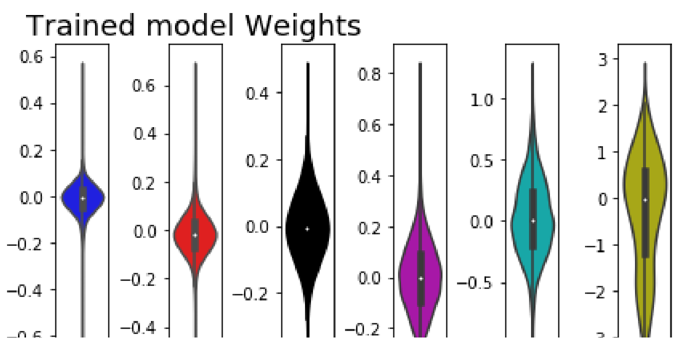
```
(784, 420)
(420,)
(420, 300)
(300,)
(300, 200)
(200,)
(200, 120)
(120,)
(120, 20)
(20,)
(20, 10)
(10,)
```

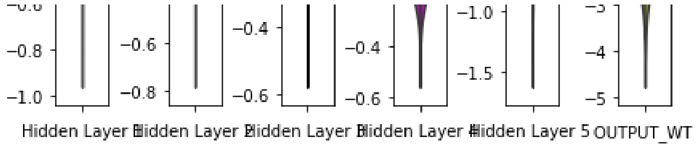
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
H3_WT = MODEL_WT[4].flatten().reshape(-1,1)
H4_WT = MODEL_WT[6].flatten().reshape(-1,1)
H5_WT = MODEL_WT[8].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[10].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4,axes5,axes6) = plt.subplots(nrows=1, ncols=6)
fig.tight_layout()

plt.subplot(1, 6, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 6, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 6, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 6, 4)
VIOLIN = SNS.violinplot(y=H4_WT, color='m')
plt.xlabel('Hidden Layer 4 ')

plt.subplot(1, 6, 5)
VIOLIN = SNS.violinplot(y=H5_WT, color='c')
plt.xlabel('Hidden Layer 5 ')


plt.subplot(1, 6, 6)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```





MODEL:4

Activation Function: relu

Optimizer: Adamax

Wt initializer : glorot_uniform

MLP Architecture: [340-240-140-100-50]

Dropout= 0.3

BatchNormalization

In [0]:

```
from keras.initializers import glorot_uniform
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
M= Sequential()
M.add(Dense(340,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=glorot_uniform
(seed=None)))
M.add(BatchNormalization())
M.add(Dropout(0.3))
M.add(Dense(240,activation='relu',kernel_initializer=glorot_uniform(seed=None)))
M.add(BatchNormalization())
M.add(Dropout(0.3))
M.add(Dense(140,activation='relu',kernel_initializer=glorot_uniform(seed=None)))
M.add(BatchNormalization())
M.add(Dropout(0.3))
M.add(Dense(100,activation='relu',kernel_initializer=glorot_uniform(seed=None)))
M.add(BatchNormalization())
M.add(Dropout(0.3))
M.add(Dense(50,activation='relu',kernel_initializer=glorot_uniform(seed=None)))
M.add(BatchNormalization())
M.add(Dropout(0.3))

M.add(Dense(10,activation='softmax'))
```

In [0]:

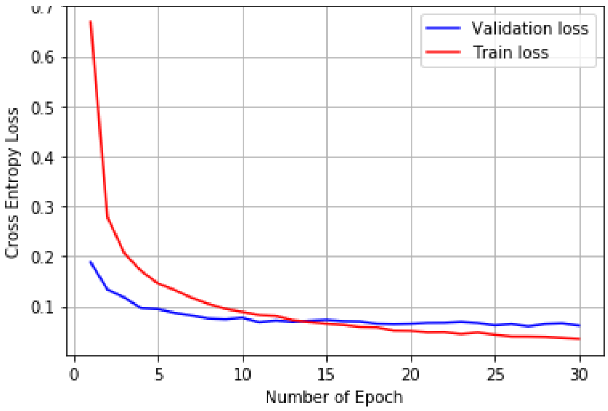
```
H=NN(M,'Adamax')
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 17s 292us/step - loss: 0.6684 - acc: 0.7974 - val_l
oss: 0.1879 - val_acc: 0.9445
Epoch 2/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.2780 - acc: 0.9225 - val_l
oss: 0.1333 - val_acc: 0.9614
Epoch 3/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.2061 - acc: 0.9436 - val_l
oss: 0.1174 - val_acc: 0.9667
Epoch 4/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.1705 - acc: 0.9533 - val_l
oss: 0.0962 - val_acc: 0.9733
Epoch 5/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.1455 - acc: 0.9607 - val_l
oss: 0.0943 - val_acc: 0.9737
Epoch 6/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.1323 - acc: 0.9636 - val_l
oss: 0.0866 - val_acc: 0.9763
Epoch 7/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.1169 - acc: 0.9679 - val_l
oss: 0.0814 - val_acc: 0.9783
Epoch 8/30
```

```
60000/60000 [==============================] - 10s 172us/step - loss: 0.1047 - acc: 0.9709 - val_l
oss: 0.0755 - val_acc: 0.9801
Epoch 9/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0949 - acc: 0.9740 - val_l
oss: 0.0739 - val_acc: 0.9797
Epoch 10/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.0887 - acc: 0.9759 - val_l
oss: 0.0771 - val_acc: 0.9800
Epoch 11/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0824 - acc: 0.9768 - val_l
oss: 0.0680 - val_acc: 0.9808
Epoch 12/30
60000/60000 [==============================] - 10s 174us/step - loss: 0.0806 - acc: 0.9776 - val_l
oss: 0.0713 - val_acc: 0.9818
Epoch 13/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0723 - acc: 0.9793 - val_l
oss: 0.0687 - val_acc: 0.9813
Epoch 14/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.0682 - acc: 0.9814 - val_l
oss: 0.0710 - val_acc: 0.9827
Epoch 15/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0648 - acc: 0.9821 - val_l
oss: 0.0727 - val_acc: 0.9817
Epoch 16/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.0626 - acc: 0.9824 - val_l
oss: 0.0700 - val_acc: 0.9817
Epoch 17/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0583 - acc: 0.9830 - val_l
oss: 0.0694 - val_acc: 0.9821
Epoch 18/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0576 - acc: 0.9832 - val_l
oss: 0.0649 - val_acc: 0.9836
Epoch 19/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0512 - acc: 0.9850 - val_l
oss: 0.0640 - val_acc: 0.9842
Epoch 20/30
60000/60000 [==============================] - 11s 176us/step - loss: 0.0507 - acc: 0.9856 - val_l
oss: 0.0647 - val_acc: 0.9839
Epoch 21/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0478 - acc: 0.9862 - val_l
oss: 0.0666 - val_acc: 0.9843
Epoch 22/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.0481 - acc: 0.9862 - val_l
oss: 0.0669 - val_acc: 0.9844
Epoch 23/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0443 - acc: 0.9872 - val_l
oss: 0.0688 - val_acc: 0.9840
Epoch 24/30
60000/60000 [==============================] - 10s 173us/step - loss: 0.0474 - acc: 0.9865 - val_l
oss: 0.0663 - val_acc: 0.9842
Epoch 25/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0429 - acc: 0.9878 - val_l
oss: 0.0618 - val_acc: 0.9846
Epoch 26/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0394 - acc: 0.9889 - val_l
oss: 0.0644 - val_acc: 0.9844
Epoch 27/30
60000/60000 [==============================] - 10s 170us/step - loss: 0.0392 - acc: 0.9885 - val_l
oss: 0.0597 - val_acc: 0.9850
Epoch 28/30
60000/60000 [==============================] - 10s 171us/step - loss: 0.0386 - acc: 0.9890 - val_l
oss: 0.0646 - val_acc: 0.9852
Epoch 29/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.0366 - acc: 0.9896 - val_l
oss: 0.0658 - val_acc: 0.9853
Epoch 30/30
60000/60000 [==============================] - 10s 172us/step - loss: 0.0346 - acc: 0.9899 - val_l
oss: 0.0613 - val_acc: 0.9847
Test score: 0.061320845727506096
Test accuracy: 0.9847
```

In [0]:

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 340)
(340,)
(340,)
(340,)
(340,)
(340,)
(340, 240)
(240,)
(240,)
(240,)
(240,)
(240,)
(240, 140)
(140,)
(140,)
(140,)
(140,)
(140,)
(140, 100)
(100,)
(100,)
(100,)
(100,)
(100,)
(100, 50)
(50,)
(50,)
(50,)
(50,)
(50,)
(50, 10)
(10,)
```

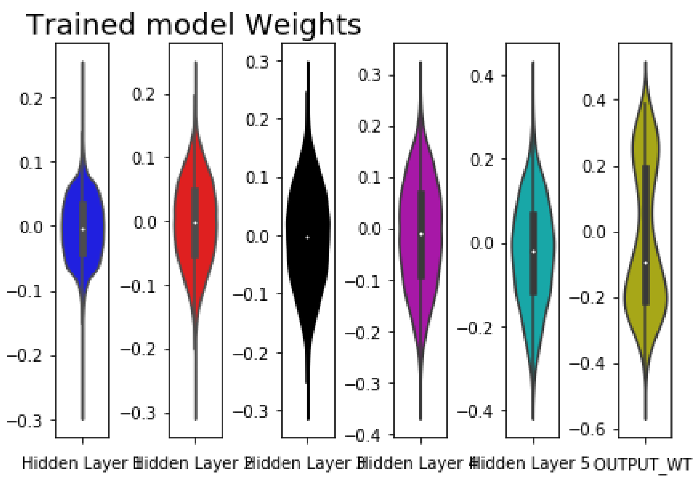
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
H4_WT = MODEL_WT[18].flatten().reshape(-1,1)
H5_WT = MODEL_WT[24].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[30].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4,axes5,axes6) = plt.subplots(nrows=1, ncols=6)
fig.tight_layout()

plt.subplot(1, 6, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 6, 2)
plt.title("Trained model Weights",size=18)
```

```
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 6, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 6, 4)
VIOLIN = SNS.violinplot(y=H4_WT, color='m')
plt.xlabel('Hidden Layer 4 ')

plt.subplot(1, 6, 5)
VIOLIN = SNS.violinplot(y=H5_WT, color='c')
plt.xlabel('Hidden Layer 5 ')


plt.subplot(1, 6, 6)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



Model :5

Activation: Relu

Optimizer: Adam

Wt initializer: All ones

Dropout rate = 0.2

MLP layers = [350-250-100]

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.initializers import Ones
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
M= Sequential()
M.add(Dense(350,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=Ones()))
M.add(Dropout(0.2))
M.add(Dense(250,activation='relu',kernel_initializer=Ones()))
M.add(Dropout(0.2))
M.add(Dense(150,activation='relu',kernel_initializer=Ones()))
M.add(Dropout(0.2))


M.add(Dense(10,activation='softmax'))
```

In [0]:

```
H = NN(M,'adam')
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 14s 240us/step - loss: 14.5332 - acc: 0.0983 - val_
loss: 14.5353 - val_acc: 0.0982
Epoch 2/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5503 - acc: 0.0973 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 3/30
60000/60000 [==============================] - 8s 134us/step - loss: 14.5396 - acc: 0.0979 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 4/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5589 - acc: 0.0967 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 5/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5563 - acc: 0.0969 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 6/30
60000/60000 [==============================] - 8s 135us/step - loss: 14.5541 - acc: 0.0970 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 7/30
60000/60000 [==============================] - 8s 134us/step - loss: 14.5458 - acc: 0.0975 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 8/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5635 - acc: 0.0965 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 9/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5426 - acc: 0.0977 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 10/30
60000/60000 [==============================] - 8s 138us/step - loss: 14.5471 - acc: 0.0975 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 11/30
60000/60000 [==============================] - 8s 137us/step - loss: 14.5407 - acc: 0.0979 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 12/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5477 - acc: 0.0974 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 13/30
60000/60000 [==============================] - 8s 138us/step - loss: 14.5517 - acc: 0.0972 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 14/30
60000/60000 [==============================] - 8s 138us/step - loss: 14.5426 - acc: 0.0977 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 15/30
60000/60000 [==============================] - 8s 135us/step - loss: 14.5579 - acc: 0.0968 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 16/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5458 - acc: 0.0975 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 17/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5552 - acc: 0.0970 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 18/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.5294 - acc: 0.0986 -
val_loss: 14.5353 - val_acc: 0.0982
Epoch 19/30
60000/60000 [==============================] - 8s 137us/step - loss: 14.6103 - acc: 0.0936 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 20/30
60000/60000 [==============================] - 8s 135us/step - loss: 14.6632 - acc: 0.0903 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 21/30
60000/60000 [==============================] - 8s 135us/step - loss: 14.6634 - acc: 0.0903 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 22/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.6610 - acc: 0.0904 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 23/30
60000/60000 [==============================] - 9s 144us/step - loss: 14.6624 - acc: 0.0903 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 24/30
60000/60000 [==============================] - 9s 142us/step - loss: 14.6597 - acc: 0.0905 -
val_loss: 14.6804 - val_acc: 0.0892
```

```
val_loss: 14.6804 - val_acc: 0.0892
Epoch 25/30
60000/60000 [==============================] - 8s 135us/step - loss: 14.6621 - acc: 0.0903 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 26/30
60000/60000 [==============================] - 8s 135us/step - loss: 14.6599 - acc: 0.0905 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 27/30
60000/60000 [==============================] - 8s 137us/step - loss: 14.6605 - acc: 0.0904 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 28/30
60000/60000 [==============================] - 8s 138us/step - loss: 14.6621 - acc: 0.0903 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 29/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.6610 - acc: 0.0904 -
val_loss: 14.6804 - val_acc: 0.0892
Epoch 30/30
60000/60000 [==============================] - 8s 136us/step - loss: 14.6597 - acc: 0.0905 -
val_loss: 14.6804 - val_acc: 0.0892
Test score: 14.680361064147949
Test accuracy: 0.0892
```

In [0]:

```
PLOT(H)
```

In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 350)
(350,)
(350, 250)
(250,)
(250, 150)
(150,)
(150, 10)
(10,)
```

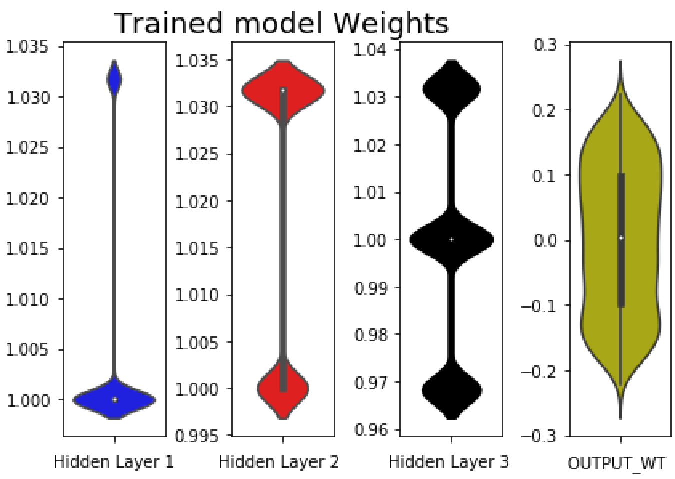
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
H3_WT = MODEL_WT[4].flatten().reshape(-1,1)
OUT_WT = MODEL_WT[6].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
```

```
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')



plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



MODEL 5.1

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.initializers import Ones
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
M= Sequential()
M.add(Dense(350,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.2))
M.add(Dense(250,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.2))
M.add(Dense(150,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.2))


M.add(Dense(10,activation='softmax'))
```

In [0]:

```
H = NN(M,'adam')
```

```
W0828 10:54:26.671952 140462218803072 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/optimizers.py:790: The name tf.train.Optimizer is dep
recated. Please use tf.compat.v1.train.Optimizer instead.

W0828 10:54:26.706413 140462218803072 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3295: The name tf.log i
s deprecated. Please use tf.math.log instead.
```

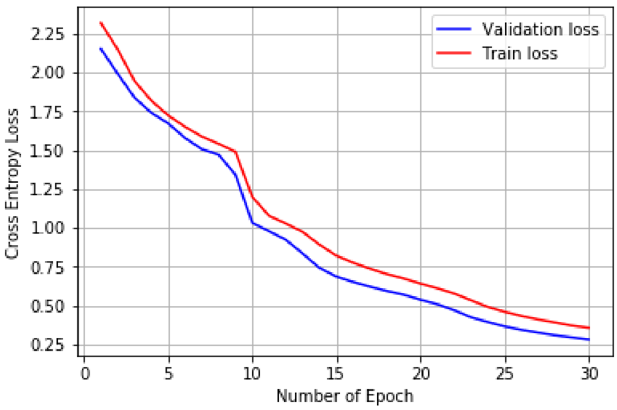
```
W0828 10:54:26.819462 140462218803072 deprecation.py:323] From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 11s 176us/step - loss: 2.3187 - acc: 0.1555 - val_l
oss: 2.1514 - val_acc: 0.1941
Epoch 2/30
60000/60000 [==============================] - 9s 145us/step - loss: 2.1470 - acc: 0.1900 -
val_loss: 1.9909 - val_acc: 0.2050
Epoch 3/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.9447 - acc: 0.2326 -
val_loss: 1.8374 - val_acc: 0.2541
Epoch 4/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.8182 - acc: 0.2568 -
val_loss: 1.7403 - val_acc: 0.2702
Epoch 5/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.7240 - acc: 0.2903 -
val_loss: 1.6723 - val_acc: 0.3270
Epoch 6/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.6499 - acc: 0.3308 -
val_loss: 1.5777 - val_acc: 0.4039
Epoch 7/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.5871 - acc: 0.3769 -
val_loss: 1.5059 - val_acc: 0.4180
Epoch 8/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.5402 - acc: 0.4028 -
val_loss: 1.4706 - val_acc: 0.4195
Epoch 9/30
60000/60000 [==============================] - 9s 146us/step - loss: 1.4897 - acc: 0.4326 -
val_loss: 1.3421 - val_acc: 0.5338
Epoch 10/30
60000/60000 [==============================] - 9s 146us/step - loss: 1.1984 - acc: 0.5644 -
val_loss: 1.0317 - val_acc: 0.6358
Epoch 11/30
60000/60000 [==============================] - 9s 150us/step - loss: 1.0765 - acc: 0.6067 -
val_loss: 0.9774 - val_acc: 0.6460
Epoch 12/30
60000/60000 [==============================] - 9s 147us/step - loss: 1.0271 - acc: 0.6266 -
val_loss: 0.9221 - val_acc: 0.6765
Epoch 13/30
60000/60000 [==============================] - 9s 147us/step - loss: 0.9732 - acc: 0.6519 -
val_loss: 0.8338 - val_acc: 0.7256
Epoch 14/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.8904 - acc: 0.6915 -
val_loss: 0.7410 - val_acc: 0.7549
Epoch 15/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.8201 - acc: 0.7158 -
val_loss: 0.6854 - val_acc: 0.7713
Epoch 16/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.7751 - acc: 0.7307 -
val_loss: 0.6506 - val_acc: 0.7782
Epoch 17/30
60000/60000 [==============================] - 9s 144us/step - loss: 0.7364 - acc: 0.7433 -
val_loss: 0.6206 - val_acc: 0.7904
Epoch 18/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.7017 - acc: 0.7580 -
val_loss: 0.5925 - val_acc: 0.8001
Epoch 19/30
60000/60000 [==============================] - 9s 144us/step - loss: 0.6734 - acc: 0.7706 -
val_loss: 0.5690 - val_acc: 0.8132
Epoch 20/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.6404 - acc: 0.7867 -
val_loss: 0.5365 - val_acc: 0.8306
Epoch 21/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.6102 - acc: 0.8014 -
val_loss: 0.5078 - val_acc: 0.8442
Epoch 22/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.5766 - acc: 0.8182 -
val_loss: 0.4693 - val_acc: 0.8578
Epoch 23/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.5331 - acc: 0.8403 -
val_loss: 0.4237 - val_acc: 0.8752
Epoch 24/30
```

```
Epoch 24/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.4904 - acc: 0.8561 -
val_loss: 0.3931 - val_acc: 0.8876
Epoch 25/30
60000/60000 [==============================] - 9s 147us/step - loss: 0.4590 - acc: 0.8663 -
val_loss: 0.3651 - val_acc: 0.8962
Epoch 26/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.4321 - acc: 0.8729 -
val_loss: 0.3429 - val_acc: 0.9009
Epoch 27/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.4095 - acc: 0.8794 -
val_loss: 0.3247 - val_acc: 0.9053
Epoch 28/30
60000/60000 [==============================] - 9s 147us/step - loss: 0.3902 - acc: 0.8858 -
val_loss: 0.3074 - val_acc: 0.9112
Epoch 29/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.3702 - acc: 0.8907 -
val_loss: 0.2939 - val_acc: 0.9155
Epoch 30/30
60000/60000 [==============================] - 9s 147us/step - loss: 0.3557 - acc: 0.8949 -
val_loss: 0.2804 - val_acc: 0.9176
Test score: 0.28041057830750943
Test accuracy: 0.9176
```

In [0]:

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

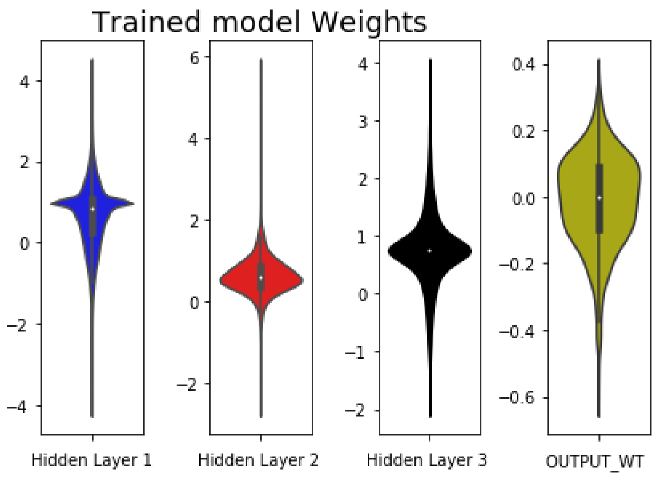
```
(784, 350)
(350,)
(350,)
(350,)
(350,)
(350,)
(350, 250)
(250,)
(250,)
(250,)
(250,)
(250,)
(250, 150)
(150,)
(150,)
(150,)
(150,)
(150,)
(150, 10)
(10,)
```

In [0]:

```
import seaborn as SNS
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
OUT_WT = MODEL_WT[18].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')



plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



In [0]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.initializers import Ones
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
M= Sequential()
M.add(Dense(350,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=Ones()))
M.add(BatchNormalization())

M.add(Dense(250,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())

M.add(Dense(150,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())



M.add(Dense(10,activation='softmax'))
```

In [0]:

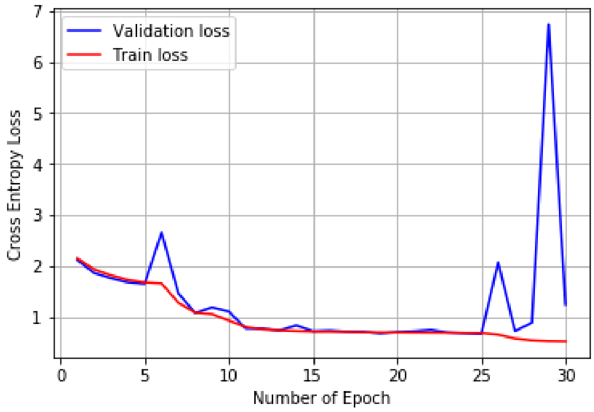
```python
H=NN(M,'adam')
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 10s 168us/step - loss: 2.1484 - acc: 0.1942 - val_l
oss: 2.1150 - val_acc: 0.1730
Epoch 2/30
60000/60000 [==============================] - 9s 144us/step - loss: 1.9294 - acc: 0.2551 -
val_loss: 1.8593 - val_acc: 0.2725
Epoch 3/30
60000/60000 [==============================] - 9s 143us/step - loss: 1.8147 - acc: 0.2835 -
val_loss: 1.7578 - val_acc: 0.2879
Epoch 4/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.7249 - acc: 0.3105 -
val_loss: 1.6762 - val_acc: 0.3297
Epoch 5/30
60000/60000 [==============================] - 9s 145us/step - loss: 1.6743 - acc: 0.3305 -
val_loss: 1.6466 - val_acc: 0.3454
Epoch 6/30
60000/60000 [==============================] - 9s 144us/step - loss: 1.6547 - acc: 0.3450 -
val_loss: 2.6546 - val_acc: 0.1159
Epoch 7/30
60000/60000 [==============================] - 9s 144us/step - loss: 1.2749 - acc: 0.5346 -
val_loss: 1.4604 - val_acc: 0.4707
Epoch 8/30
60000/60000 [==============================] - 9s 144us/step - loss: 1.0833 - acc: 0.6144 -
val_loss: 1.0761 - val_acc: 0.6224
Epoch 9/30
60000/60000 [==============================] - 9s 143us/step - loss: 1.0524 - acc: 0.6264 -
val_loss: 1.1858 - val_acc: 0.5428
Epoch 10/30
60000/60000 [==============================] - 9s 144us/step - loss: 0.9257 - acc: 0.6857 -
val_loss: 1.1035 - val_acc: 0.6119
Epoch 11/30
60000/60000 [==============================] - 9s 144us/step - loss: 0.8017 - acc: 0.7387 -
val_loss: 0.7716 - val_acc: 0.7482
Epoch 12/30
60000/60000 [==============================] - 9s 148us/step - loss: 0.7540 - acc: 0.7552 -
val_loss: 0.7698 - val_acc: 0.7548
Epoch 13/30
60000/60000 [==============================] - 9s 150us/step - loss: 0.7346 - acc: 0.7618 -
val_loss: 0.7322 - val_acc: 0.7594
Epoch 14/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.7220 - acc: 0.7677 -
val_loss: 0.8329 - val_acc: 0.7251
Epoch 15/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.7108 - acc: 0.7732 -
val_loss: 0.7183 - val_acc: 0.7783
Epoch 16/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.7078 - acc: 0.7728 -
val_loss: 0.7299 - val_acc: 0.7695
Epoch 17/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.7066 - acc: 0.7739 -
val_loss: 0.7082 - val_acc: 0.7810
Epoch 18/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.6981 - acc: 0.7758 -
val_loss: 0.7047 - val_acc: 0.7753
Epoch 19/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.6948 - acc: 0.7781 -
val_loss: 0.6731 - val_acc: 0.7908
Epoch 20/30
60000/60000 [==============================] - 9s 147us/step - loss: 0.6925 - acc: 0.7771 -
val_loss: 0.7004 - val_acc: 0.7757
Epoch 21/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.6904 - acc: 0.7776 -
val_loss: 0.7167 - val_acc: 0.7720
Epoch 22/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.6886 - acc: 0.7796 -
val_loss: 0.7427 - val_acc: 0.7556
Epoch 23/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.6870 - acc: 0.7807 -
val_loss: 0.6919 - val_acc: 0.7806
Epoch 24/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.6843 - acc: 0.7823 -
val_loss: 0.6771 - val_acc: 0.7889
Epoch 25/30
```

```
60000/60000 [==============================] - 9s 146us/step - loss: 0.6832 - acc: 0.7810 -
val_loss: 0.6655 - val_acc: 0.7944
Epoch 26/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.6490 - acc: 0.7978 -
val_loss: 2.0686 - val_acc: 0.4639
Epoch 27/30
60000/60000 [==============================] - 9s 144us/step - loss: 0.5736 - acc: 0.8263 -
val_loss: 0.7213 - val_acc: 0.7806
Epoch 28/30
60000/60000 [==============================] - 9s 145us/step - loss: 0.5372 - acc: 0.8388 -
val_loss: 0.8845 - val_acc: 0.7050
Epoch 29/30
60000/60000 [==============================] - 9s 146us/step - loss: 0.5227 - acc: 0.8452 -
val_loss: 6.7350 - val_acc: 0.2115
Epoch 30/30
60000/60000 [==============================] - 9s 147us/step - loss: 0.5177 - acc: 0.8471 -
val_loss: 1.2408 - val_acc: 0.6169
Test score: 1.2407595457077025
Test accuracy: 0.6169
```

In [0]:

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 350)
(350,)
(350,)
(350,)
(350,)
(350,)
(350, 250)
(250,)
(250,)
(250,)
(250,)
(250,)
(250, 150)
(150,)
(150,)
(150,)
(150,)
(150,)
(150, 10)
(10,)
```

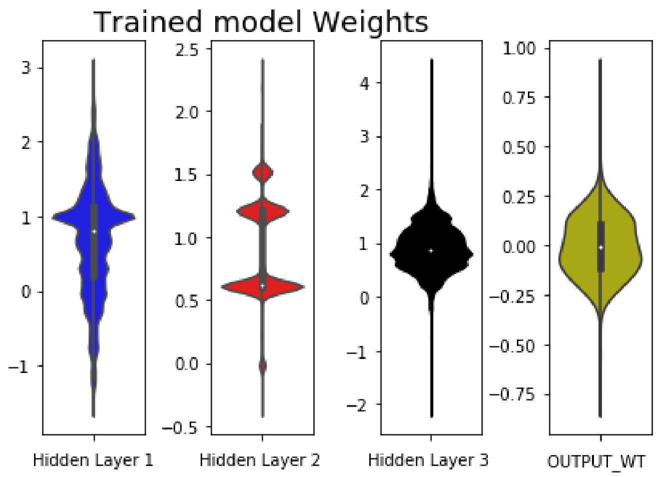
In [0]:

```
import seaborn as SNS
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
```

```
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
OUT_WT = MODEL_WT[18].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')



plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



Model :6

Activation: Relu

Optimizer: Adam

Wt initializer: All ones

Dropout rate = 0.4

MLP layers = [450-350-250-150-50]

BatchNormalization

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.initializers import Ones
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
M= Sequential()
M.add(Dense(450,activation='relu',input_shape=(X_TRAIN.shape[1],),kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.4))
```

```
M.add(Dense(350,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.4))
M.add(Dense(250,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.4))
M.add(Dense(150,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.4))
M.add(Dense(50,activation='relu',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.4))

M.add(Dense(10,activation='softmax'))
```

In [0]:

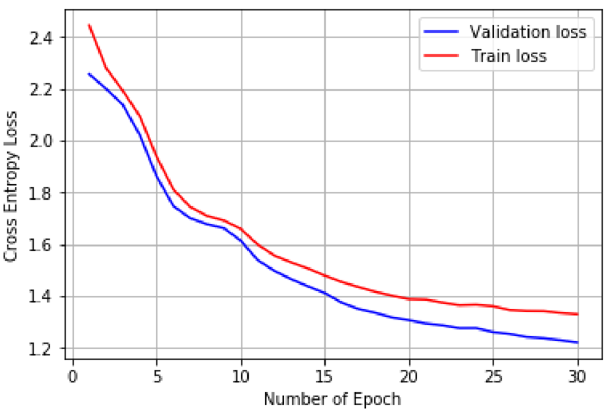
```
H=NN(M,'adam')
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 23s 378us/step - loss: 2.4447 - acc: 0.1321 - val_l
oss: 2.2568 - val_acc: 0.1632
Epoch 2/30
60000/60000 [==============================] - 15s 249us/step - loss: 2.2809 - acc: 0.1513 - val_l
oss: 2.2004 - val_acc: 0.1787
Epoch 3/30
60000/60000 [==============================] - 15s 250us/step - loss: 2.1925 - acc: 0.1762 - val_l
oss: 2.1385 - val_acc: 0.2012
Epoch 4/30
60000/60000 [==============================] - 15s 250us/step - loss: 2.0951 - acc: 0.2074 - val_l
oss: 2.0226 - val_acc: 0.2097
Epoch 5/30
60000/60000 [==============================] - 15s 249us/step - loss: 1.9392 - acc: 0.2347 - val_l
oss: 1.8634 - val_acc: 0.2311
Epoch 6/30
60000/60000 [==============================] - 24s 404us/step - loss: 1.8124 - acc: 0.2504 - val_l
oss: 1.7479 - val_acc: 0.2686
Epoch 7/30
60000/60000 [==============================] - 15s 253us/step - loss: 1.7443 - acc: 0.2677 - val_l
oss: 1.7014 - val_acc: 0.2766
Epoch 8/30
60000/60000 [==============================] - 15s 249us/step - loss: 1.7103 - acc: 0.2819 - val_l
oss: 1.6779 - val_acc: 0.3058
Epoch 9/30
60000/60000 [==============================] - 15s 248us/step - loss: 1.6924 - acc: 0.2968 - val_l
oss: 1.6628 - val_acc: 0.3540
Epoch 10/30
60000/60000 [==============================] - 15s 250us/step - loss: 1.6605 - acc: 0.3268 - val_l
oss: 1.6150 - val_acc: 0.3631
Epoch 11/30
60000/60000 [==============================] - 15s 248us/step - loss: 1.6000 - acc: 0.3521 - val_l
oss: 1.5403 - val_acc: 0.3827
Epoch 12/30
60000/60000 [==============================] - 15s 249us/step - loss: 1.5570 - acc: 0.3715 - val_l
oss: 1.4981 - val_acc: 0.4031
Epoch 13/30
60000/60000 [==============================] - 15s 247us/step - loss: 1.5309 - acc: 0.3852 - val_l
oss: 1.4661 - val_acc: 0.4366
Epoch 14/30
60000/60000 [==============================] - 15s 248us/step - loss: 1.5072 - acc: 0.4028 - val_l
oss: 1.4382 - val_acc: 0.4503
Epoch 15/30
60000/60000 [==============================] - 15s 248us/step - loss: 1.4797 - acc: 0.4168 - val_l
oss: 1.4122 - val_acc: 0.4575
Epoch 16/30
60000/60000 [==============================] - 15s 253us/step - loss: 1.4552 - acc: 0.4325 - val_l
oss: 1.3755 - val_acc: 0.5183
Epoch 17/30
60000/60000 [==============================] - 15s 249us/step - loss: 1.4356 - acc: 0.4380 - val_l
oss: 1.3512 - val_acc: 0.5144
Epoch 18/30
60000/60000 [==============================] - 15s 249us/step - loss: 1.4171 - acc: 0.4438 - val_l
oss: 1.3362 - val_acc: 0.5196
Epoch 19/30
```

```
60000/60000 [==============================] - 15s 253us/step - loss: 1.4019 - acc: 0.4533 - val_l
oss: 1.3182 - val_acc: 0.5367
Epoch 20/30
60000/60000 [==============================] - 15s 250us/step - loss: 1.3889 - acc: 0.4586 - val_l
oss: 1.3077 - val_acc: 0.5494
Epoch 21/30
60000/60000 [==============================] - 15s 243us/step - loss: 1.3872 - acc: 0.4602 - val_l
oss: 1.2950 - val_acc: 0.5541
Epoch 22/30
60000/60000 [==============================] - 15s 242us/step - loss: 1.3752 - acc: 0.4655 - val_l
oss: 1.2878 - val_acc: 0.5643
Epoch 23/30
60000/60000 [==============================] - 15s 251us/step - loss: 1.3655 - acc: 0.4685 - val_l
oss: 1.2768 - val_acc: 0.5622
Epoch 24/30
60000/60000 [==============================] - 15s 244us/step - loss: 1.3673 - acc: 0.4644 - val_l
oss: 1.2770 - val_acc: 0.5810
Epoch 25/30
60000/60000 [==============================] - 15s 243us/step - loss: 1.3616 - acc: 0.4676 - val_l
oss: 1.2617 - val_acc: 0.5729
Epoch 26/30
60000/60000 [==============================] - 15s 245us/step - loss: 1.3469 - acc: 0.4792 - val_l
oss: 1.2545 - val_acc: 0.5799
Epoch 27/30
60000/60000 [==============================] - 15s 248us/step - loss: 1.3433 - acc: 0.4815 - val_l
oss: 1.2426 - val_acc: 0.5928
Epoch 28/30
60000/60000 [==============================] - 14s 240us/step - loss: 1.3427 - acc: 0.4836 - val_l
oss: 1.2378 - val_acc: 0.5974
Epoch 29/30
60000/60000 [==============================] - 15s 244us/step - loss: 1.3357 - acc: 0.4836 - val_l
oss: 1.2304 - val_acc: 0.5730
Epoch 30/30
60000/60000 [==============================] - 15s 245us/step - loss: 1.3310 - acc: 0.4840 - val_l
oss: 1.2219 - val_acc: 0.6061
Test score: 1.2218715141296386
Test accuracy: 0.6061
```

In [0]:

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

```
(784, 450)
(450,)
(450,)
(450,)
(450,)
(450,)
(450, 350)
(350,)
(350,)
(350,)
```

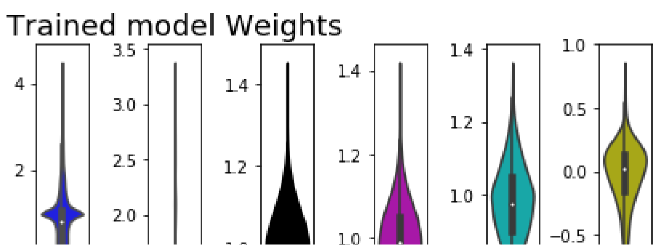
```
(350,)
(350,)
(350, 250)
(250,)
(250,)
(250,)
(250,)
(250,)
(250, 150)
(150,)
(150,)
(150,)
(150,)
(150,)
(150, 50)
(50,)
(50,)
(50,)
(50,)
(50,)
(50, 10)
(10,)
```

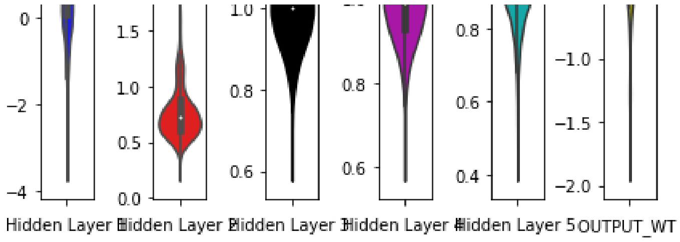
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
H4_WT = MODEL_WT[18].flatten().reshape(-1,1)
H5_WT = MODEL_WT[24].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[30].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4,axes5,axes6) = plt.subplots(nrows=1, ncols=6)
fig.tight_layout()

plt.subplot(1, 6, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 6, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 6, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')

plt.subplot(1, 6, 4)
VIOLIN = SNS.violinplot(y=H4_WT, color='m')
plt.xlabel('Hidden Layer 4 ')

plt.subplot(1, 6, 5)
VIOLIN = SNS.violinplot(y=H5_WT, color='c')
plt.xlabel('Hidden Layer 5 ')


plt.subplot(1, 6, 6)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```





Model :7

Activation: SoftPlus

Optimizer: Adadelta

Wt initializer: All ones

Dropout rate = 0.5

MLP layers = [350-250-100]

BatchNormalization

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.initializers import Ones
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
M= Sequential()
M.add(Dense(260,activation='softplus',input_shape=(X_TRAIN.shape[1],),kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.5))
M.add(Dense(140,activation='softplus',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.5))
M.add(Dense(40,activation='softplus',kernel_initializer=Ones()))
M.add(BatchNormalization())
M.add(Dropout(0.5))


M.add(Dense(10,activation='softmax'))
```

In [0]:

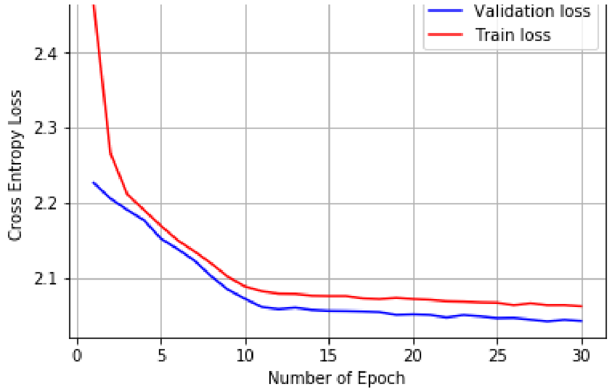
```
H = NN(M,'Adadelta')
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 15s 252us/step - loss: 2.4643 - acc: 0.1397 - val_l
oss: 2.2264 - val_acc: 0.1720
Epoch 2/30
60000/60000 [==============================] - 9s 151us/step - loss: 2.2659 - acc: 0.1589 -
val_loss: 2.2057 - val_acc: 0.1782
Epoch 3/30
60000/60000 [==============================] - 7s 122us/step - loss: 2.2114 - acc: 0.1761 -
val_loss: 2.1905 - val_acc: 0.1820
Epoch 4/30
60000/60000 [==============================] - 7s 123us/step - loss: 2.1901 - acc: 0.1852 -
val_loss: 2.1765 - val_acc: 0.1910
Epoch 5/30
60000/60000 [==============================] - 8s 126us/step - loss: 2.1692 - acc: 0.1976 -
val_loss: 2.1522 - val_acc: 0.2011
Epoch 6/30
60000/60000 [==============================] - 8s 128us/step - loss: 2.1500 - acc: 0.2054 -
val_loss: 2.1383 - val_acc: 0.2053
Epoch 7/30
60000/60000 [==============================] - 8s 129us/step - loss: 2.1351 - acc: 0.2135 -
val_loss: 2.1228 - val_acc: 0.2118
Epoch 8/30
60000/60000 [==============================] - 8s 126us/step - loss: 2.1190 - acc: 0.2185 -
```

```
val_loss: 2.1018 - val_acc: 0.2181
Epoch 9/30
60000/60000 [==============================] - 7s 120us/step - loss: 2.1010 - acc: 0.2225 -
val_loss: 2.0841 - val_acc: 0.2237
Epoch 10/30
60000/60000 [==============================] - 7s 123us/step - loss: 2.0885 - acc: 0.2261 -
val_loss: 2.0721 - val_acc: 0.2219
Epoch 11/30
60000/60000 [==============================] - 8s 131us/step - loss: 2.0821 - acc: 0.2255 -
val_loss: 2.0610 - val_acc: 0.2246
Epoch 12/30
60000/60000 [==============================] - 7s 124us/step - loss: 2.0788 - acc: 0.2268 -
val_loss: 2.0581 - val_acc: 0.2275
Epoch 13/30
60000/60000 [==============================] - 8s 126us/step - loss: 2.0785 - acc: 0.2257 -
val_loss: 2.0601 - val_acc: 0.2249
Epoch 14/30
60000/60000 [==============================] - 7s 125us/step - loss: 2.0761 - acc: 0.2254 -
val_loss: 2.0570 - val_acc: 0.2221
Epoch 15/30
60000/60000 [==============================] - 8s 126us/step - loss: 2.0757 - acc: 0.2250 -
val_loss: 2.0559 - val_acc: 0.2254
Epoch 16/30
60000/60000 [==============================] - 8s 128us/step - loss: 2.0757 - acc: 0.2270 -
val_loss: 2.0556 - val_acc: 0.2274
Epoch 17/30
60000/60000 [==============================] - 8s 127us/step - loss: 2.0725 - acc: 0.2265 -
val_loss: 2.0551 - val_acc: 0.2265
Epoch 18/30
60000/60000 [==============================] - 8s 125us/step - loss: 2.0715 - acc: 0.2293 -
val_loss: 2.0543 - val_acc: 0.2277
Epoch 19/30
60000/60000 [==============================] - 7s 123us/step - loss: 2.0732 - acc: 0.2278 -
val_loss: 2.0504 - val_acc: 0.2282
Epoch 20/30
60000/60000 [==============================] - 7s 122us/step - loss: 2.0716 - acc: 0.2298 -
val_loss: 2.0512 - val_acc: 0.2280
Epoch 21/30
60000/60000 [==============================] - 7s 122us/step - loss: 2.0707 - acc: 0.2285 -
val_loss: 2.0504 - val_acc: 0.2282
Epoch 22/30
60000/60000 [==============================] - 7s 122us/step - loss: 2.0687 - acc: 0.2290 -
val_loss: 2.0470 - val_acc: 0.2272
Epoch 23/30
60000/60000 [==============================] - 7s 122us/step - loss: 2.0681 - acc: 0.2294 -
val_loss: 2.0503 - val_acc: 0.2320
Epoch 24/30
60000/60000 [==============================] - 7s 123us/step - loss: 2.0672 - acc: 0.2284 -
val_loss: 2.0485 - val_acc: 0.2290
Epoch 25/30
60000/60000 [==============================] - 7s 122us/step - loss: 2.0668 - acc: 0.2284 -
val_loss: 2.0460 - val_acc: 0.2290
Epoch 26/30
60000/60000 [==============================] - 7s 124us/step - loss: 2.0635 - acc: 0.2303 -
val_loss: 2.0464 - val_acc: 0.2267
Epoch 27/30
60000/60000 [==============================] - 7s 123us/step - loss: 2.0661 - acc: 0.2303 -
val_loss: 2.0441 - val_acc: 0.2295
Epoch 28/30
60000/60000 [==============================] - 7s 123us/step - loss: 2.0635 - acc: 0.2316 -
val_loss: 2.0418 - val_acc: 0.2273
Epoch 29/30
60000/60000 [==============================] - 7s 123us/step - loss: 2.0635 - acc: 0.2320 -
val_loss: 2.0440 - val_acc: 0.2342
Epoch 30/30
60000/60000 [==============================] - 7s 118us/step - loss: 2.0620 - acc: 0.2326 -
val_loss: 2.0424 - val_acc: 0.2305
Test score: 2.0424131748199463
Test accuracy: 0.2305
```

In [0]:

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

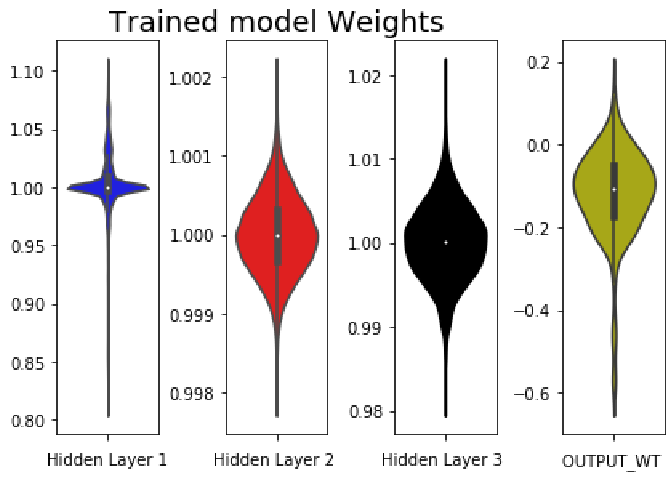
```
(784, 260)
(260,)
(260,)
(260,)
(260,)
(260,)
(260, 140)
(140,)
(140,)
(140,)
(140,)
(140,)
(140, 40)
(40,)
(40,)
(40,)
(40,)
(40,)
(40, 10)
(10,)
```

In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[6].flatten().reshape(-1,1)
H3_WT = MODEL_WT[12].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[18].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')



plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```



Model :8

Activation: Softplus

Optimizer: Adadelta

Wt initializer: All ones

MLP layers = [260-140-40]

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.initializers import Ones
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
M= Sequential()
M.add(Dense(260,activation='softplus',input_shape=(X_TRAIN.shape[1],),kernel_initializer=Ones()))

M.add(Dense(140,activation='softplus',kernel_initializer=Ones()))

M.add(Dense(40,activation='softplus',kernel_initializer=Ones()))



M.add(Dense(10,activation='softmax'))
```

In [0]:

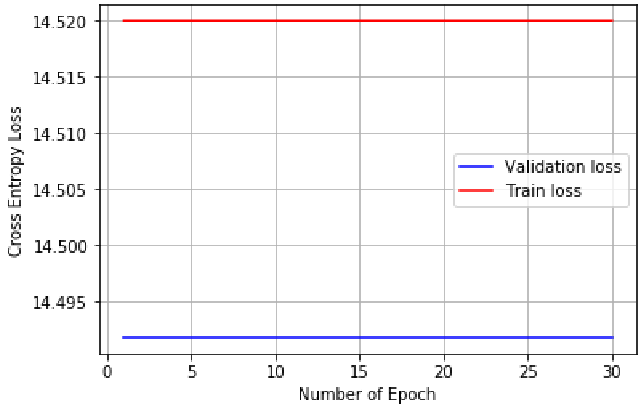
```
H = NN(M,'Adadelta')
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/30
60000/60000 [==============================] - 14s 230us/step - loss: 14.5200 - acc: 0.0991 - val_
loss: 14.4918 - val_acc: 0.1009
Epoch 2/30
60000/60000 [==============================] - 6s 93us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 3/30
60000/60000 [==============================] - 6s 94us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 4/30
60000/60000 [==============================] - 7s 111us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 5/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 6/30
60000/60000 [==============================] - 6s 100us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
```

```
Epoch 7/30
60000/60000 [==============================] - 6s 97us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 8/30
60000/60000 [==============================] - 6s 97us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 9/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 10/30
60000/60000 [==============================] - 6s 97us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 11/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 12/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 13/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 14/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 15/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 16/30
60000/60000 [==============================] - 6s 94us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 17/30
60000/60000 [==============================] - 6s 94us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 18/30
60000/60000 [==============================] - 6s 93us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 19/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 20/30
60000/60000 [==============================] - 6s 94us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 21/30
60000/60000 [==============================] - 6s 96us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 22/30
60000/60000 [==============================] - 6s 98us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 23/30
60000/60000 [==============================] - 6s 95us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 24/30
60000/60000 [==============================] - 6s 93us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 25/30
60000/60000 [==============================] - 6s 94us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 26/30
60000/60000 [==============================] - 6s 96us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 27/30
60000/60000 [==============================] - 6s 104us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 28/30
60000/60000 [==============================] - 6s 100us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 29/30
60000/60000 [==============================] - 6s 100us/step - loss: 14.5200 - acc: 0.0991 -
val_loss: 14.4918 - val_acc: 0.1009
Epoch 30/30
60000/60000 [==============================] - 6s 99us/step - loss: 14.5200 - acc: 0.0992 -
val_loss: 14.4918 - val_acc: 0.1009
Test score: 14.491779391479492
Test accuracy: 0.1009
```

```
In [0]:
```

```
PLOT(H)
```



In [0]:

```
MODEL_WT = M.get_weights()
for i in range(0,len(MODEL_WT)):
    print(MODEL_WT[i].shape)
```

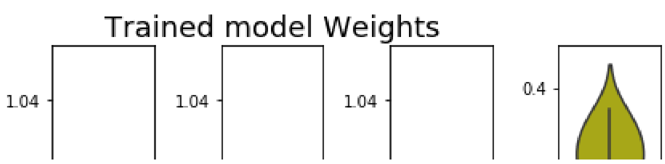
```
(784, 260)
(260,)
(260, 140)
(140,)
(140, 40)
(40,)
(40, 10)
(10,)
```

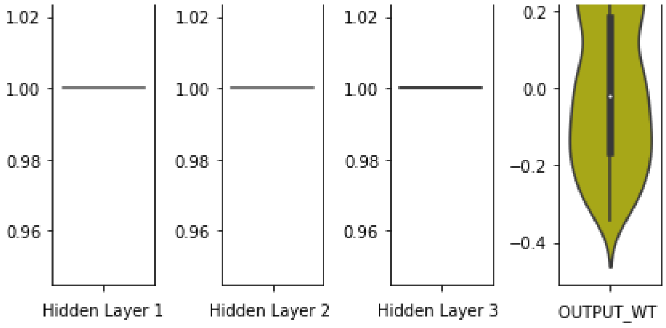
In [0]:

```
H1_WT = MODEL_WT[0].flatten().reshape(-1,1)
H2_WT = MODEL_WT[2].flatten().reshape(-1,1)
H3_WT = MODEL_WT[4].flatten().reshape(-1,1)
OUT_WT= MODEL_WT[6].flatten().reshape(-1,1)


fig,(axes1,axes2,axes3,axes4) = plt.subplots(nrows=1, ncols=4)
fig.tight_layout()

plt.subplot(1, 4, 1)
VIOLIN = SNS.violinplot(y=H1_WT,color='b')
plt.xlabel('Hidden Layer 1')

plt.subplot(1, 4, 2)
plt.title("Trained model Weights",size=18)
VIOLIN = SNS.violinplot(y=H2_WT, color='r')
plt.xlabel('Hidden Layer 2 ')

plt.subplot(1, 4, 3)
VIOLIN = SNS.violinplot(y=H3_WT, color='k')
plt.xlabel('Hidden Layer 3 ')


plt.subplot(1, 4, 4)
VIOLIN = SNS.violinplot(y=OUT_WT, color='y')
plt.xlabel('OUTPUT_WT ')
```

Out[0]:

```
Text(0.5, 0, 'OUTPUT_WT ')
```





In [18]:

```
from prettytable import PrettyTable
X=PrettyTable()
print(" "*40+"CONCLUSION")
print("="*100)
X.field_names = ["Model","Number Of Hidden Layers","Neurons in Layer",'Dropout rate',"Test Loss"]
X.add_row(["Sigmoid-SGD-RandomUniform","3","[240-120-60]",'No Dropout',0.719])
X.add_row(["Tanh+Adagrad+RandomNormal","3","[340-150-20]",'No Dropout',0.064])
X.add_row(["Hard Sigmoid+AdaDelta+Glorot Normal+Batch Normalization+Dropout","5","[420-300-200-120
-20]",0.3,0.12])

X.add_row(["Relu + Adamax +glorot Uniform +BN+Dropout","5","[340-240-140-100-50]",0.3,0.07])

X.add_row(["Relu+Adam+All Ones +Dropout","3","[350-250-100]",'0.2',14.54])
X.add_row(["Relu+Adam+All Ones +BN+DROPOUT","3","[350-250-100]",'0.2',0.28])
X.add_row(["Relu+Adam+All Ones +BN","3","[350-250-100]",'No',1.24])
X.add_row(["Relu+Adam+All Ones+Batch Normalization+Dropout","5","[450-350-250-150-50]",'0.4',1.22]
)

X.add_row(["softPlus+Adadelta+All One+BN+Dropout","3","[260-140-40]",0.5,2.04])
X.add_row(["Softplus+Adadelta+AllOne ","3","[350-250-100]","No Dropout",14.46])
print(X)
```

```
                                        CONCLUSION
====================================================================================================

+------------------------------------------------------------------+--------------------------+------
------------+--------------+-----------+
|                              Model                               | Number Of Hidden Layers |    Neu
ns in Layer    | Dropout rate | Test Loss |
+------------------------------------------------------------------+--------------------------+------
------------+--------------+-----------+
|                    Sigmoid-SGD-RandomUniform                     |            3             |      [
0-120-60]      |  No Dropout  |   0.719   |
|                    Tanh+Adagrad+RandomNormal                     |            3             |      [
0-150-20]      |  No Dropout  |   0.064   |
| Hard Sigmoid+AdaDelta+Glorot Normal+Batch Normalization+Dropout |            5             | [420-
300-200-120-20] |     0.3      |    0.12   |
|            Relu + Adamax +glorot Uniform +BN+Dropout             |            5             | [340-
40-140-100-50] |     0.3      |    0.07   |
|                   Relu+Adam+All Ones +Dropout                    |            3             |     [3
-250-100]      |     0.2      |   14.54   |
|                  Relu+Adam+All Ones +BN+DROPOUT                  |            3             |     [3
-250-100]      |     0.2      |    0.28   |
|                      Relu+Adam+All Ones +BN                      |            3             |
[350-250-100]      |      No      |    1.24   |
|          Relu+Adam+All Ones+Batch Normalization+Dropout          |            5             | [450-
50-250-150-50] |     0.4      |    1.22   |
|               softPlus+Adadelta+All One+BN+Dropout               |            3             |      [
0-140-40]      |     0.5      |    2.04   |
|                    Softplus+Adadelta+AllOne                      |            3             |     [3
-250-100]      |  No Dropout  |   14.46   |
+------------------------------------------------------------------+--------------------------+------
------------+--------------+-----------+
```

1. In first 9 model I have use Relu as activation function ,adam as optimizer and intialize weight using HE_Normal,I have tried out different architecture with different layer and ploted Train loss/Validation loss vs epoch to check my model perfomance.

- If Train loss is too high than validation loss we say that Model is Underfit.
- If Train Loss is too low than validation loss then Model is Overfit

- If Train Loss is too low than validation loss then Model is Overfit.
- What we want in Our Model is that both this loss should be low and approximatly close to each other.
- It is clearly seen from first preety table that doing Batch Normalization followed by dropout of particular droprate improve model perfomance , i.e Test loss is decreasing as we add do Batch Normalization and add dropout.
- I have also plotted Violin plot of every model to see weight distribution . Thumb rule is that initial Weight should not be all zero or all large numbers this can create problem of vanishing or exploidig gradient.
- One thing to be noted that if we want to extract those updated weight from our model we must pay special attention to number of neurons we are using in each layer and also take care that whether we are doing Batch Normalization and dropout.

Ex.

```
let say we have 3  hidden layer and we are not using any BN and dropout, if we extract weig
ht using MODEL.get_weights() then we will be getting 8 vectors of shape depending on number
of neurons we used.

If  we have 3  hidden layer and if  use any BN and dropout, if we extract weight using
MODEL.get_weights() then we will be getting 20 vectors of shape depending on number of neurc
ns we used.
```

1. In Next Eight Model(with differnet architecture)I have tried with different optimizer,activation function and used different weight intializer.
2. Best Model in my case is giving Test Loss of 0.07 it has 5 hidden layers[340-240-140-100-50] Relu + Adamax +glorot Uniform +BN+Dropout.

7.What I observe is using BN+Dropout improve our model perfomance, I have also tried out model with only BN not dropout it is giving more test loss in comparison, Most of the time scenario will get worse if we only use dropout without doing BN it will give very high Test loss.

8.When I use 3Hidden layer Model{Last Model} [Softplus+Adadelta+AllOne(Weight Intializer)] with no dropout and BN ,there is an no update of weight,weight distribution is constant and Training loss and Validation loss both are not changing as Number of epoch increases.

1. When I add BN and dropout in same model ,it perfomance improved significantly.