CNN Archietectures on MNIST dataset

In [0]:

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [3]:

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

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

In [4]:

```
X_train.shape
```

Out[4]:

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

In [5]:

```
# input image dimensions
img_rows, img_cols = X_train.shape[1] , X_train.shape[2]
num_pixels = img_rows*img_cols
print(num_pixels)
```

```
784
```

In [0]:

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

In [0]:

```
#Normalizing
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train/255.0
X_test = X_test/255.0
```

In [12]:

```
print(X_train.shape)
print(X_test.shape)
print(input_shape)
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
(28, 28, 1)
```

In [0]:

```
batch_size = 128
num_classes = 10
epochs = 15
```

In [0]:

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

1. Archietecture -1 -

1.1.1 Without droput, Batch Normalization

In [13]:

```
model_1_a = Sequential()

#conv layer 1
model_1_a.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=input_shape))
model_1_a.add(MaxPooling2D(pool_size=(2,2), padding='same'))

#conv layer 2
model_1_a.add(Conv2D(64, kernel_size=(3,3), activation='relu', input_shape=input_shape))

#conv layer 3
model_1_a.add(Conv2D(128, kernel_size=(3,3), activation='relu', input_shape=input_shape))
model_1_a.add(MaxPooling2D(pool_size=(2,2), padding='valid'))

#Flatten
model_1_a.add(Flatten())

#output class
model_1_a.add(Dense(num_classes, activation='softmax'))

#compile with adam optimizers
model_1_a.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#summary
model_1_a.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Plea
se use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please us
e tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Pleas
e use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is deprecated. Please u
se tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name t
f.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is deprecated. Please use tf.ma
th.log instead.

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496     
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 9, 9, 128)         73856     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 10)                20490     
=================================================================
Total params: 113,162
Trainable params: 113,162
Non-trainable params: 0
_________________________________________________________________
```

In [14]:

```
#fitting the model
history_1_a = model_1_a.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size=batch_si
ze, epochs=epochs, verbose=1)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/math_grad.py:1424: where (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
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please us
e tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf
.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is deprecated. Please use t
f.compat.v1.Session instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/15
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session is deprecated. P
lease use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please us
e tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is deprecated. Plea
se use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initialized is
deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer is deprecated.
Please use tf.compat.v1.variables_initializer instead.

60000/60000 [==============================] - 19s 318us/step - loss: 0.2067 - acc: 0.9403 - val_l
oss: 0.0589 - val_acc: 0.9822
Epoch 2/15
```

```
60000/60000 [==============================] - 4s 59us/step - loss: 0.0593 - acc: 0.9817 -
val_loss: 0.0415 - val_acc: 0.9859
Epoch 3/15
60000/60000 [==============================] - 4s 59us/step - loss: 0.0407 - acc: 0.9872 -
val_loss: 0.0393 - val_acc: 0.9876
Epoch 4/15
60000/60000 [==============================] - 4s 59us/step - loss: 0.0315 - acc: 0.9902 -
val_loss: 0.0342 - val_acc: 0.9889
Epoch 5/15
60000/60000 [==============================] - 4s 59us/step - loss: 0.0245 - acc: 0.9922 -
val_loss: 0.0285 - val_acc: 0.9901
Epoch 6/15
60000/60000 [==============================] - 4s 60us/step - loss: 0.0203 - acc: 0.9934 -
val_loss: 0.0243 - val_acc: 0.9923
Epoch 7/15
60000/60000 [==============================] - 4s 59us/step - loss: 0.0176 - acc: 0.9943 -
val_loss: 0.0271 - val_acc: 0.9913
Epoch 8/15
60000/60000 [==============================] - 4s 59us/step - loss: 0.0139 - acc: 0.9953 -
val_loss: 0.0240 - val_acc: 0.9926
Epoch 9/15
60000/60000 [==============================] - 4s 61us/step - loss: 0.0109 - acc: 0.9966 -
val_loss: 0.0356 - val_acc: 0.9899
Epoch 10/15
60000/60000 [==============================] - 4s 59us/step - loss: 0.0090 - acc: 0.9971 -
val_loss: 0.0281 - val_acc: 0.9919
Epoch 11/15
60000/60000 [==============================] - 4s 60us/step - loss: 0.0095 - acc: 0.9968 -
val_loss: 0.0258 - val_acc: 0.9924
Epoch 12/15
60000/60000 [==============================] - 4s 65us/step - loss: 0.0080 - acc: 0.9972 -
val_loss: 0.0308 - val_acc: 0.9909
Epoch 13/15
60000/60000 [==============================] - 4s 62us/step - loss: 0.0071 - acc: 0.9976 -
val_loss: 0.0303 - val_acc: 0.9919
Epoch 14/15
60000/60000 [==============================] - 4s 66us/step - loss: 0.0053 - acc: 0.9981 -
val_loss: 0.0395 - val_acc: 0.9898
Epoch 15/15
60000/60000 [==============================] - 4s 65us/step - loss: 0.0057 - acc: 0.9980 -
val_loss: 0.0358 - val_acc: 0.9907
```

In [15]:

```python
score = model_1_a.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy:', score[1])
```

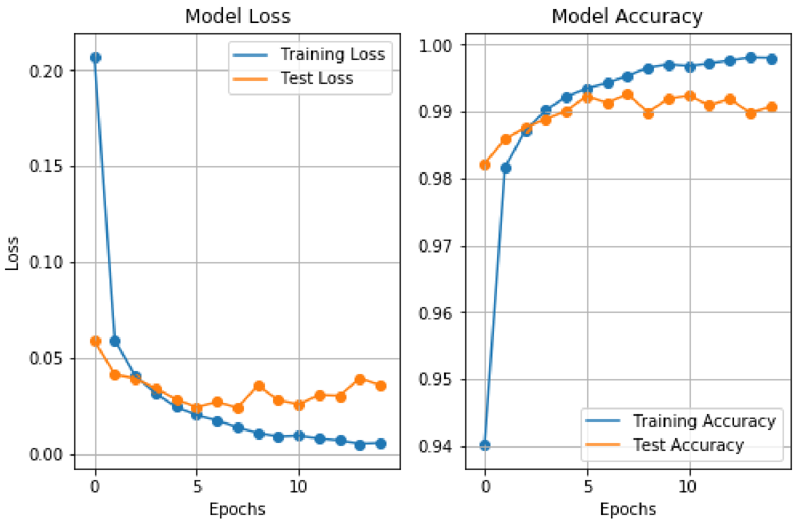
```
Test Accuracy: 0.9907
```

1.1.2 Plotting Epoch vs Loss, Epoch vs Accuracy

In [48]:

```python
plt.figure(figsize=(8,5))

plt.subplot(1,2,1)
plt.plot(history_1_a.history['loss'], label='Training Loss')
plt.plot(history_1_a.history['val_loss'], label='Test Loss')
plt.scatter([i for i in range(epochs)], history_1_a.history['loss'])
plt.scatter([i for i in range(epochs)], history_1_a.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()

plt.subplot(1,2,2)
plt.plot(history_1_a.history['acc'], label='Training Accuracy')
plt.plot(history_1_a.history['val_acc'], label='Test Accuracy')
plt.scatter([i for i in range(epochs)], history_1_a.history['acc'])
plt.scatter([i for i in range(epochs)], history_1_a.history['val_acc'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.grid()
```

```
plt.grid()
plt.legend()

plt.show()
```



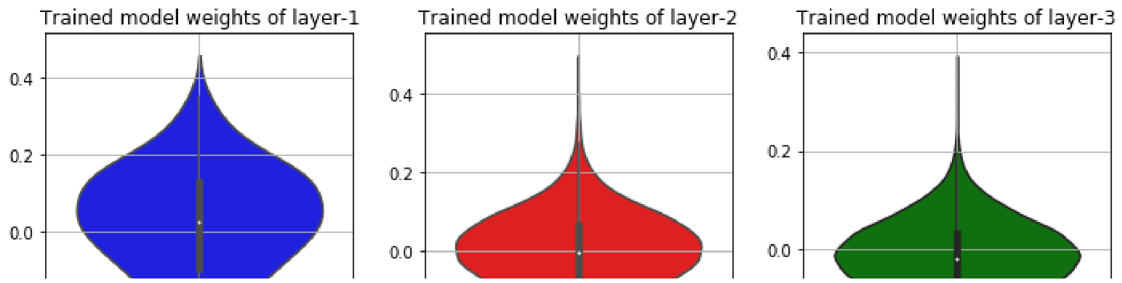
1.1.3 Distribution of Weights

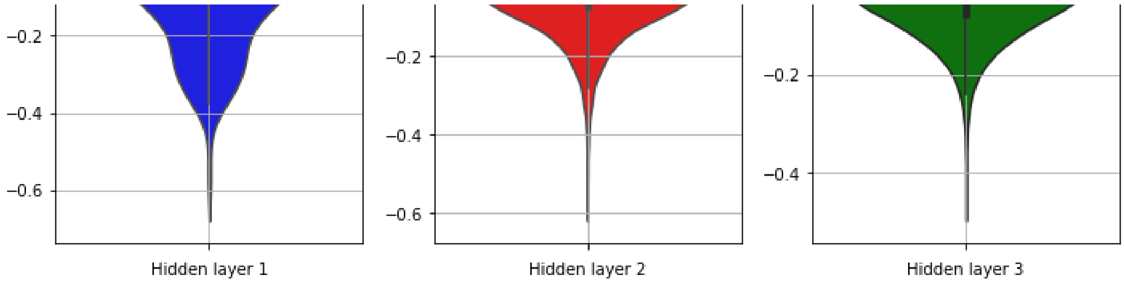
In [0]:

```
w_after_1_a = model_1_a.get_weights()
h1_w = w_after_1_a[0].flatten().reshape(-1,1)
h2_w = w_after_1_a[2].flatten().reshape(-1,1)
h3_w = w_after_1_a[4].flatten().reshape(-1,1)
```

In [19]:

```
fig = plt.figure(figsize=(10, 5))
plt.title('Weights of the model after trained')

plt.subplot(1,3,1)
plt.title('Trained model weights of layer-1')
sns.violinplot(y=h1_w, color='b')
plt.xlabel('Hidden layer 1')
plt.grid()
plt.tight_layout()

plt.subplot(1,3,2)
plt.title('Trained model weights of layer-2')
sns.violinplot(y=h2_w, color='r')
plt.xlabel('Hidden layer 2')
plt.tight_layout()
plt.grid()

plt.subplot(1,3,3)
plt.title('Trained model weights of layer-3')
sns.violinplot(y=h3_w, color='g')
plt.xlabel('Hidden layer 3')
plt.tight_layout()
plt.grid()


plt.show()
```





1.2 With Dropout

In [20]:

```
model_1_b = Sequential()

#conv layer 1 with maxpooling and dropout
model_1_b.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=input_shape))
model_1_b.add(MaxPooling2D(pool_size=(2,2), padding='valid'))
model_1_b.add(Dropout(0.5))

#conv layer 2 without maxpooling
model_1_b.add(Conv2D(64, kernel_size=(3,3), activation='relu', input_shape=input_shape))

#conv layer 3 with maxpoling and dropout
model_1_b.add(Conv2D(128, kernel_size=(3,3), activation='relu', input_shape=input_shape))
model_1_b.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model_1_b.add(Dropout(0.5))

#Flatten
model_1_b.add(Flatten())

#output class
model_1_b.add(Dense(num_classes, activation='softmax'))

#compile with adam optimizers
model_1_b.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#summary
model_1_b.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name tf.placeholder_with_default is
deprecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future
version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 9, 9, 128)         73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                32010
=================================================================
Total params: 124,682
```

```
Trainable params: 124,682
Non-trainable params: 0
_________________________________________________________________
```

In [21]:

```
history_1_b = model_1_b.fit(X_train, y_train, validation_data=(X_test, y_test), verbose=1, batch_si
ze=batch_size, epochs=epochs)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 5s 77us/step - loss: 0.2814 - acc: 0.9122 -
val_loss: 0.0570 - val_acc: 0.9822
Epoch 2/15
60000/60000 [==============================] - 4s 69us/step - loss: 0.0874 - acc: 0.9729 -
val_loss: 0.0393 - val_acc: 0.9876
Epoch 3/15
60000/60000 [==============================] - 4s 68us/step - loss: 0.0637 - acc: 0.9805 -
val_loss: 0.0319 - val_acc: 0.9890
Epoch 4/15
60000/60000 [==============================] - 4s 68us/step - loss: 0.0534 - acc: 0.9833 -
val_loss: 0.0263 - val_acc: 0.9912
Epoch 5/15
60000/60000 [==============================] - 4s 68us/step - loss: 0.0456 - acc: 0.9853 -
val_loss: 0.0267 - val_acc: 0.9909
Epoch 6/15
60000/60000 [==============================] - 4s 67us/step - loss: 0.0410 - acc: 0.9871 -
val_loss: 0.0228 - val_acc: 0.9921
Epoch 7/15
60000/60000 [==============================] - 4s 69us/step - loss: 0.0382 - acc: 0.9877 -
val_loss: 0.0210 - val_acc: 0.9928
Epoch 8/15
60000/60000 [==============================] - 4s 67us/step - loss: 0.0348 - acc: 0.9886 -
val_loss: 0.0200 - val_acc: 0.9934
Epoch 9/15
60000/60000 [==============================] - 4s 66us/step - loss: 0.0320 - acc: 0.9898 -
val_loss: 0.0196 - val_acc: 0.9937
Epoch 10/15
60000/60000 [==============================] - 4s 64us/step - loss: 0.0300 - acc: 0.9902 -
val_loss: 0.0204 - val_acc: 0.9936
Epoch 11/15
60000/60000 [==============================] - 4s 64us/step - loss: 0.0285 - acc: 0.9906 -
val_loss: 0.0210 - val_acc: 0.9932
Epoch 12/15
60000/60000 [==============================] - 4s 65us/step - loss: 0.0266 - acc: 0.9914 -
val_loss: 0.0198 - val_acc: 0.9935
Epoch 13/15
60000/60000 [==============================] - 4s 64us/step - loss: 0.0241 - acc: 0.9923 -
val_loss: 0.0200 - val_acc: 0.9938
Epoch 14/15
60000/60000 [==============================] - 4s 65us/step - loss: 0.0223 - acc: 0.9929 -
val_loss: 0.0184 - val_acc: 0.9944
Epoch 15/15
60000/60000 [==============================] - 4s 65us/step - loss: 0.0228 - acc: 0.9926 -
val_loss: 0.0197 - val_acc: 0.9940
```

In [22]:

```
score_1_b = model_1_b.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy with Dropout:', score_1_b[1])
```

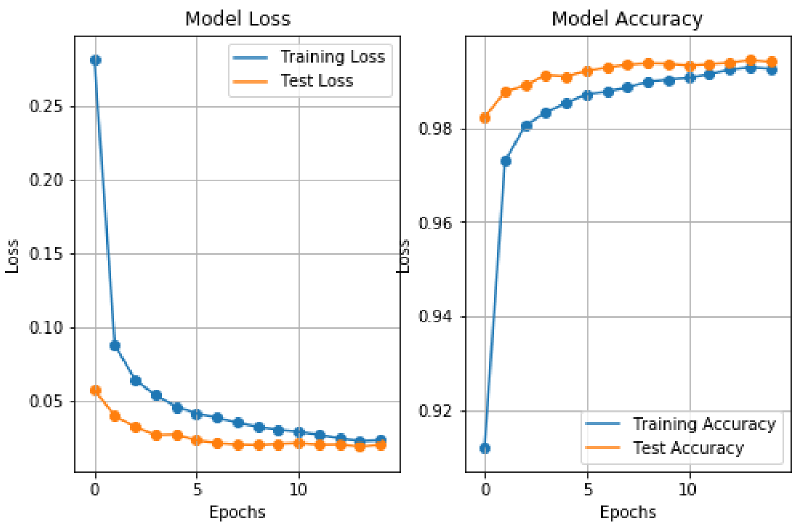
```
Test Accuracy with Dropout: 0.994
```

1.2.2 Plotting Epoch vs Loss , Epoch vs Accuracy

In [47]:

```
fig = plt.figure(figsize=(8, 5))

plt.subplot(1,2,1)
plt.plot(history_1_b.history['loss'], label='Training Loss')
```

```
plt.plot(history_1_b.history['val_loss'], label='Test Loss')
plt.scatter([i for i in range(epochs)], history_1_b.history['loss'])
plt.scatter([i for i in range(epochs)], history_1_b.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()

plt.subplot(1,2,2)
plt.plot(history_1_b.history['acc'], label='Training Accuracy')
plt.plot(history_1_b.history['val_acc'], label='Test Accuracy')
plt.scatter([i for i in range(epochs)], history_1_b.history['acc'])
plt.scatter([i for i in range(epochs)], history_1_b.history['val_acc'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()

plt.show()
```



1.2.3 Distribution of Weights

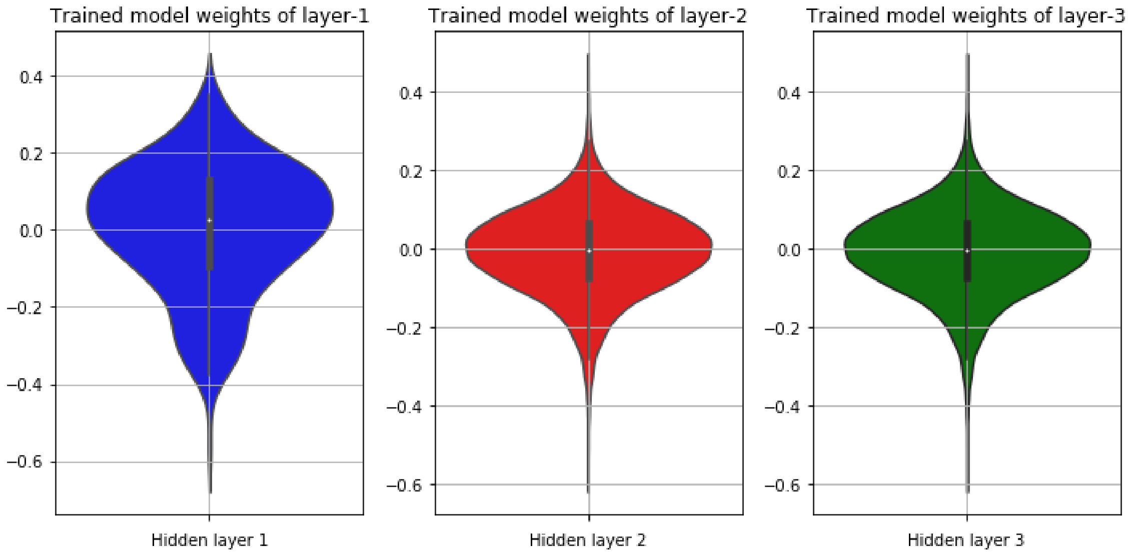
In [0]:

```
w_after_1_b = model_1_b.get_weights()
h1_w = w_after_1_b[0].flatten().reshape(-1,1)
h2_w = w_after_1_b[2].flatten().reshape(-1,1)
h3_w = w_after_1_b[4].flatten().reshape(-1,1)
```

In [26]:

```
fig = plt.figure(figsize=(10, 5))
#plt.suptitle('Weights of the model after trained')

plt.subplot(1,3,1)
plt.title('Trained model weights of layer-1')
sns.violinplot(y=h1_w, color='b')
plt.xlabel('Hidden layer 1')
plt.grid()
plt.tight_layout()

plt.subplot(1,3,2)
plt.title('Trained model weights of layer-2')
sns.violinplot(y=h2_w, color='r')
plt.xlabel('Hidden layer 2')
plt.tight_layout()
plt.grid()

plt.subplot(1,3,3)
plt.title('Trained model weights of layer-3')
sns.violinplot(y=h2_w, color='g')
plt.xlabel('Hidden layer 3')
```

```
plt.xlabel('Hidden layer 3')
plt.tight_layout()
plt.grid()

plt.show()
```



2. Architecture - 2

2.1.1 Without Dropout

In [33]:

```
model_2_a = Sequential()

#conv layer 1
model_2_a.add(Conv2D(128, kernel_size=(2,2), activation='relu', input_shape=input_shape))

#conv layer 2
model_2_a.add(Conv2D(256, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_2_a.add(MaxPooling2D(pool_size=(2,2), padding='same'))

#conv layer 3
model_2_a.add(Conv2D(512, kernel_size=(2,2), activation='relu', input_shape=input_shape))

#conv layer 4
model_2_a.add(Conv2D(256, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_2_a.add(MaxPooling2D(pool_size=(2,2), padding='same'))

#conv layer 5
model_2_a.add(Conv2D(128, kernel_size=(2,2), activation='relu', input_shape=input_shape))


#Flatten
model_2_a.add(Flatten())

#output class
model_2_a.add(Dense(num_classes, activation='softmax'))

#compile with adam optimizers
model_2_a.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#summary
model_2_a.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_32 (Conv2D)           (None, 27, 27, 128)       640       
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 26, 26, 256)       131328    
```

```
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 13, 13, 256)       0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 12, 12, 512)       524800
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 11, 11, 256)       524544
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 5, 5, 128)         131200
_________________________________________________________________
flatten_3 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                32010
=================================================================
Total params: 1,344,522
Trainable params: 1,344,522
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
history_2_a = model_2_a.fit(X_train, y_train, validation_data=(X_test, y_test), verbose=1, batch_si
ze=batch_size, epochs=epochs)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 33s 542us/step - loss: 0.1465 - acc: 0.9538 - val_l
oss: 0.0390 - val_acc: 0.9878
Epoch 2/15
60000/60000 [==============================] - 30s 494us/step - loss: 0.0424 - acc: 0.9870 - val_l
oss: 0.0389 - val_acc: 0.9886
Epoch 3/15
60000/60000 [==============================] - 30s 494us/step - loss: 0.0302 - acc: 0.9901 - val_l
oss: 0.0301 - val_acc: 0.9909
Epoch 4/15
60000/60000 [==============================] - 30s 497us/step - loss: 0.0229 - acc: 0.9927 - val_l
oss: 0.0322 - val_acc: 0.9892
Epoch 5/15
60000/60000 [==============================] - 29s 488us/step - loss: 0.0198 - acc: 0.9933 - val_l
oss: 0.0273 - val_acc: 0.9918
Epoch 6/15
60000/60000 [==============================] - 29s 482us/step - loss: 0.0145 - acc: 0.9953 - val_l
oss: 0.0297 - val_acc: 0.9913
Epoch 7/15
60000/60000 [==============================] - 29s 483us/step - loss: 0.0121 - acc: 0.9960 - val_l
oss: 0.0324 - val_acc: 0.9910
Epoch 8/15
60000/60000 [==============================] - 29s 489us/step - loss: 0.0117 - acc: 0.9957 - val_l
oss: 0.0296 - val_acc: 0.9915
Epoch 9/15
60000/60000 [==============================] - 29s 482us/step - loss: 0.0104 - acc: 0.9967 - val_l
oss: 0.0351 - val_acc: 0.9914
Epoch 10/15
60000/60000 [==============================] - 29s 487us/step - loss: 0.0082 - acc: 0.9972 - val_l
oss: 0.0315 - val_acc: 0.9917
Epoch 11/15
60000/60000 [==============================] - 29s 488us/step - loss: 0.0078 - acc: 0.9972 - val_l
oss: 0.0383 - val_acc: 0.9906
Epoch 12/15
60000/60000 [==============================] - 29s 485us/step - loss: 0.0073 - acc: 0.9978 - val_l
oss: 0.0351 - val_acc: 0.9911
Epoch 13/15
60000/60000 [==============================] - 29s 481us/step - loss: 0.0067 - acc: 0.9979 - val_l
oss: 0.0348 - val_acc: 0.9922
Epoch 14/15
60000/60000 [==============================] - 29s 479us/step - loss: 0.0057 - acc: 0.9982 - val_l
oss: 0.0462 - val_acc: 0.9904
Epoch 15/15
60000/60000 [==============================] - 29s 478us/step - loss: 0.0066 - acc: 0.9980 - val_l
oss: 0.0372 - val_acc: 0.9915
```

In [37]:

```
score_2_a = model_2_a.evaluate(X_test, y_test, verbose=0)
```

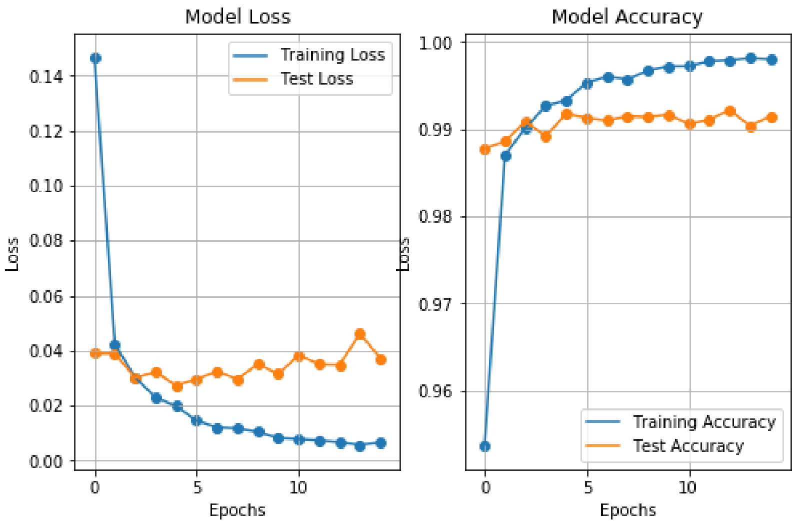
```
print('Test Accuracy without Dropout', score_2_a[1])
```

```
Test Accuracy without Dropout 0.9915
```

2.1.2 Loss vs Epoch

In [45]:

```
fig = plt.figure(figsize=(8, 5))

plt.subplot(1,2,1)
plt.plot(history_2_a.history['loss'], label='Training Loss')
plt.plot(history_2_a.history['val_loss'], label='Test Loss')
plt.scatter([i for i in range(epochs)], history_2_a.history['loss'])
plt.scatter([i for i in range(epochs)], history_2_a.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()


plt.subplot(1,2,2)
plt.plot(history_2_a.history['acc'], label='Training Accuracy')
plt.plot(history_2_a.history['val_acc'], label='Test Accuracy')
plt.scatter([i for i in range(epochs)], history_2_a.history['acc'])
plt.scatter([i for i in range(epochs)], history_2_a.history['val_acc'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()

plt.show()
```



2.1.3 Distribution of Weights

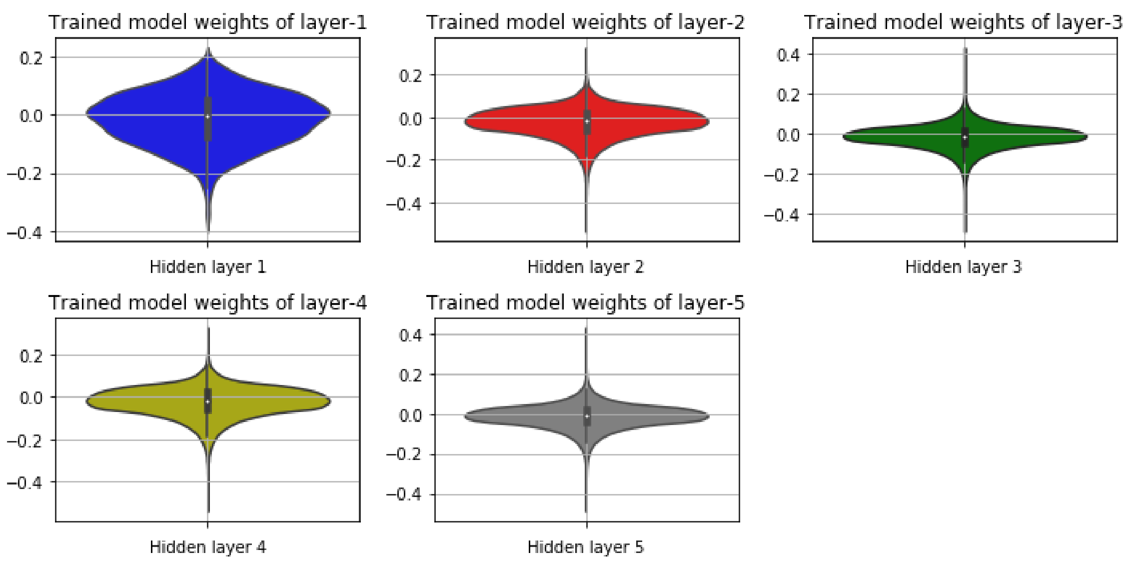
In [0]:

```
w_after_2_a = model_2_a.get_weights()
h1_w = w_after_2_a[0].flatten().reshape(-1,1)
h2_w = w_after_2_a[2].flatten().reshape(-1,1)
h3_w = w_after_2_a[4].flatten().reshape(-1,1)
h4_w = w_after_2_a[6].flatten().reshape(-1,1)
h5_w = w_after_2_a[8].flatten().reshape(-1,1)
```

In [41]:

```
fig = plt.figure(figsize=(10, 5))
plt.title('Weights of the model after trained')
```

```
plt.subplot(2,3,1)
plt.title('Trained model weights of layer-1')
sns.violinplot(y=h1_w, color='b')
plt.xlabel('Hidden layer 1')
plt.grid()
plt.tight_layout()

plt.subplot(2,3,2)
plt.title('Trained model weights of layer-2')
sns.violinplot(y=h2_w, color='r')
plt.xlabel('Hidden layer 2')
plt.tight_layout()
plt.grid()


plt.subplot(2,3,3)
plt.title('Trained model weights of layer-3')
sns.violinplot(y=h3_w, color='g')
plt.xlabel('Hidden layer 3')
plt.grid()
plt.tight_layout()

plt.subplot(2,3,4)
plt.title('Trained model weights of layer-4')
sns.violinplot(y=h2_w, color='y')
plt.xlabel('Hidden layer 4')
plt.tight_layout()
plt.grid()


plt.subplot(2,3,5)
plt.title('Trained model weights of layer-5')
sns.violinplot(y=h3_w, color='gray')
plt.xlabel('Hidden layer 5')
plt.grid()
plt.tight_layout()



plt.show()
```



2.2 With Dropout and Batch Normalization

In [49]:

```
model_2_b = Sequential()

#conv layer 1
model_2_b.add(Conv2D(32, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_2_b.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model_2_b.add(Dropout(0.5))

#conv layer 2
```

```python
model_2_b.add(Conv2D(64, kernel_size=(2,2), activation='relu', input_shape=input_shape))

#conv layer 3
model_2_b.add(Conv2D(128, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_2_b.add(MaxPooling2D(pool_size=(2,2), padding='valid'))
model_2_b.add(Dropout(0.5))

#conv layer 4
model_2_b.add(Conv2D(256, kernel_size=(2,2), activation='relu', input_shape=input_shape))

#conv layer 5
model_2_b.add(Conv2D(512, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_2_b.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model_2_b.add(Dropout(0.5))


#Flatten
model_2_b.add(Flatten())

#output class
model_2_b.add(Dense(num_classes, activation='softmax'))

#compile with adam optimizers
model_2_b.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#summary
model_2_b.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_37 (Conv2D)           (None, 27, 27, 32)        160       
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 14, 14, 32)        0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 13, 13, 64)        8256      
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 12, 12, 128)       32896     
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 6, 6, 128)         0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 6, 6, 128)         0         
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 5, 5, 256)         131328    
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 4, 4, 512)         524800    
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 2, 2, 512)         0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 2, 2, 512)         0         
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0         
_________________________________________________________________
dense_4 (Dense)              (None, 10)                20490     
=================================================================
Total params: 717,930
Trainable params: 717,930
Non-trainable params: 0
_________________________________________________________________
```

In [50]:

```python
history_2_b = model_2_b.fit(X_train, y_train, validation_data=(X_test, y_test), verbose=1, batch_si
ze=batch_size, epochs=epochs)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 7s 122us/step - loss: 0.3642 - acc: 0.8788 -
val_loss: 0.0529 - val_acc: 0.9826
Epoch 2/15
60000/60000 [==============================] - 6s 98us/step - loss: 0.0950 - acc: 0.9697 -
val_loss: 0.0373 - val_acc: 0.9870
```

```
Epoch 3/15
60000/60000 [==============================] - 6s 95us/step - loss: 0.0720 - acc: 0.9775 -
val_loss: 0.0295 - val_acc: 0.9894
Epoch 4/15
60000/60000 [==============================] - 6s 98us/step - loss: 0.0586 - acc: 0.9821 -
val_loss: 0.0304 - val_acc: 0.9889
Epoch 5/15
60000/60000 [==============================] - 6s 94us/step - loss: 0.0521 - acc: 0.9833 -
val_loss: 0.0292 - val_acc: 0.9892
Epoch 6/15
60000/60000 [==============================] - 6s 93us/step - loss: 0.0486 - acc: 0.9844 -
val_loss: 0.0255 - val_acc: 0.9906
Epoch 7/15
60000/60000 [==============================] - 6s 92us/step - loss: 0.0418 - acc: 0.9867 -
val_loss: 0.0232 - val_acc: 0.9909
Epoch 8/15
60000/60000 [==============================] - 6s 92us/step - loss: 0.0391 - acc: 0.9882 -
val_loss: 0.0238 - val_acc: 0.9917
Epoch 9/15
60000/60000 [==============================] - 6s 93us/step - loss: 0.0355 - acc: 0.9885 -
val_loss: 0.0204 - val_acc: 0.9931
Epoch 10/15
60000/60000 [==============================] - 6s 94us/step - loss: 0.0355 - acc: 0.9885 -
val_loss: 0.0204 - val_acc: 0.9932
Epoch 11/15
60000/60000 [==============================] - 6s 95us/step - loss: 0.0328 - acc: 0.9897 -
val_loss: 0.0209 - val_acc: 0.9926
Epoch 12/15
60000/60000 [==============================] - 6s 94us/step - loss: 0.0326 - acc: 0.9899 -
val_loss: 0.0211 - val_acc: 0.9933
Epoch 13/15
60000/60000 [==============================] - 6s 95us/step - loss: 0.0294 - acc: 0.9904 -
val_loss: 0.0215 - val_acc: 0.9930
Epoch 14/15
60000/60000 [==============================] - 6s 95us/step - loss: 0.0274 - acc: 0.9910 -
val_loss: 0.0205 - val_acc: 0.9933
Epoch 15/15
60000/60000 [==============================] - 6s 96us/step - loss: 0.0264 - acc: 0.9915 -
val_loss: 0.0206 - val_acc: 0.9934
```

In [51]:

```
score_2_b = model_2_b.evaluate(X_test, y_test, verbose=0)
print('Test accuracy with Dropout and BatchNorm', score_2_b[1])
```

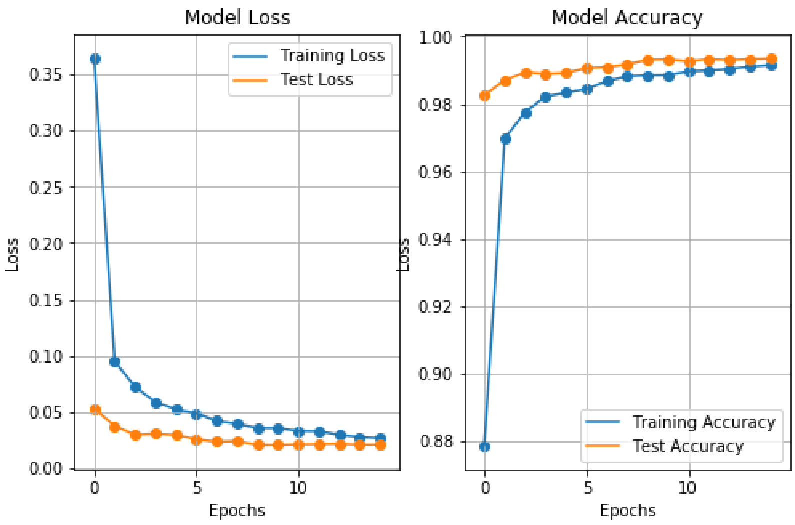
```
Test accuracy with Dropout and BatchNorm 0.9934
```

2.2.1 Epoch vs Loss, Epoch vs Accuracy

In [52]:

```
fig = plt.figure(figsize=(8,5))

plt.subplot(1,2,1)
plt.plot(history_2_b.history['loss'], label='Training Loss')
plt.plot(history_2_b.history['val_loss'], label='Test Loss')
plt.scatter([i for i in range(epochs)], history_2_b.history['loss'])
plt.scatter([i for i in range(epochs)], history_2_b.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()

plt.subplot(1,2,2)
plt.plot(history_2_b.history['acc'], label='Training Accuracy')
plt.plot(history_2_b.history['val_acc'], label='Test Accuracy')
plt.scatter([i for i in range(epochs)], history_2_b.history['acc'])
plt.scatter([i for i in range(epochs)], history_2_b.history['val_acc'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
```

```
plt.legend()

plt.show()
```



2.2.2 Distribution of Weights

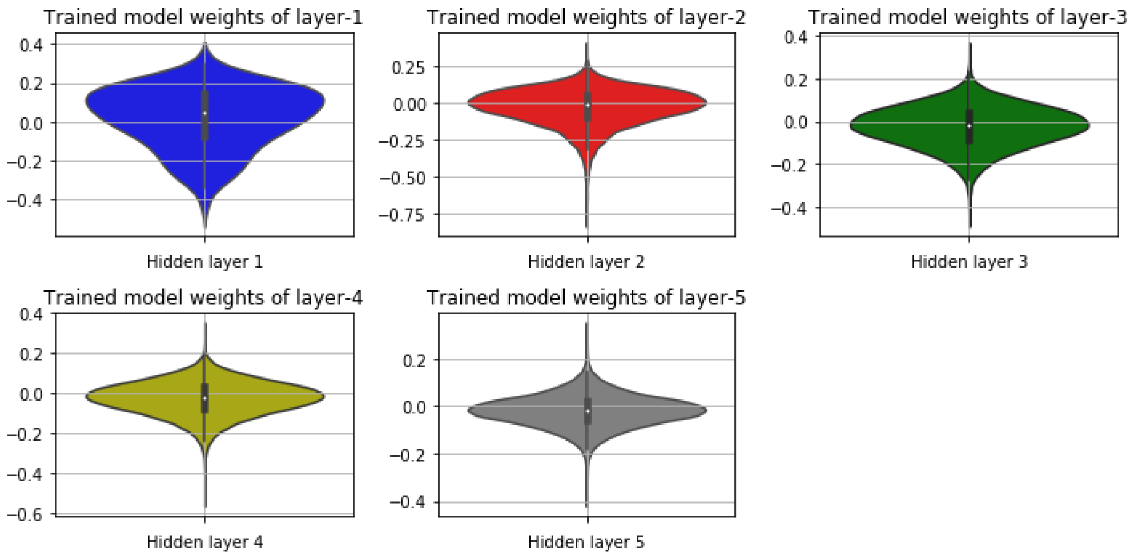
In [0]:

```
w_after_2_b = model_2_b.get_weights()
h1_w = w_after_2_b[0].flatten().reshape(-1,1)
h2_w = w_after_2_b[2].flatten().reshape(-1,1)
h3_w = w_after_2_b[4].flatten().reshape(-1,1)
h4_w = w_after_2_b[6].flatten().reshape(-1,1)
h5_w = w_after_2_b[8].flatten().reshape(-1,1)
```

In [60]:

```
fig = plt.figure(figsize=(10, 5))
plt.title('Weights of the model after trained')

plt.subplot(2,3,1)
plt.title('Trained model weights of layer-1')
sns.violinplot(y=h1_w, color='b')
plt.xlabel('Hidden layer 1')
plt.grid()
plt.tight_layout()

plt.subplot(2,3,2)
plt.title('Trained model weights of layer-2')
sns.violinplot(y=h2_w, color='r')
plt.xlabel('Hidden layer 2')
plt.tight_layout()
plt.grid()


plt.subplot(2,3,3)
plt.title('Trained model weights of layer-3')
sns.violinplot(y=h3_w, color='g')
plt.xlabel('Hidden layer 3')
plt.grid()
plt.tight_layout()

plt.subplot(2,3,4)
plt.title('Trained model weights of layer-4')
sns.violinplot(y=h4_w, color='y')
plt.xlabel('Hidden layer 4')
plt.tight_layout()
plt.grid()


plt.subplot(2,3,5)
plt.title('Trained model weights of layer-5')
sns.violinplot(y=h5_w, color='gray')
plt.xlabel('Hidden layer 5')
```

```
plt.xlabel('Hidden layer 5')
plt.grid()
plt.tight_layout()

plt.show()
```



3. Architecture - 3

3.1.1 Without Dropout

Note:

```
- Other than kernel size = (2,2) gives the Error of 'Negative Shape'.
  That's why i am using the kernel size of (2,2)
```

In [68]:

```
model_3_a = Sequential()

#conv layer 1 with maxpooling
model_3_a.add(Conv2D(64, kernel_size=(2,2), activation='relu', input_shape=input_shape))

#conv layer 2 with maxpooling
model_3_a.add(Conv2D(128, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_3_a.add(MaxPooling2D(pool_size=(2,2), padding='same'))

#conv layer 3 with maxpoling and dropout
model_3_a.add(Conv2D(256, kernel_size=(2,2), activation='relu', input_shape=input_shape))

#conv layer 4 without maxpoling and dropout
model_3_a.add(Conv2D(512, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_3_a.add(MaxPooling2D(pool_size=(2,2), padding='same'))

#conv layer 5 with maxpoling and dropout
model_3_a.add(Conv2D(256, kernel_size=(2,2), activation='relu', input_shape=input_shape))

#conv layer 6 with maxpoling and dropout
model_3_a.add(Conv2D(128, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_3_a.add(MaxPooling2D(pool_size=(2,2), padding='same'))

#conv layer 7 with maxpoling and dropout
model_3_a.add(Conv2D(64, kernel_size=(2,2), activation='relu', input_shape=input_shape))


#Flatten
model_3_a.add(Flatten())

#output class
model_3_a.add(Dense(num_classes, activation='softmax'))
```

```
#compile with adam optimizers
model_3_a.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#summary
model_3_a.summary()
```

```
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_101 (Conv2D)          (None, 27, 27, 64)        320       
_________________________________________________________________
conv2d_102 (Conv2D)          (None, 26, 26, 128)       32896     
_________________________________________________________________
max_pooling2d_47 (MaxPooling (None, 13, 13, 128)       0         
_________________________________________________________________
conv2d_103 (Conv2D)          (None, 12, 12, 256)       131328    
_________________________________________________________________
conv2d_104 (Conv2D)          (None, 11, 11, 512)       524800    
_________________________________________________________________
max_pooling2d_48 (MaxPooling (None, 6, 6, 512)         0         
_________________________________________________________________
conv2d_105 (Conv2D)          (None, 5, 5, 256)         524544    
_________________________________________________________________
conv2d_106 (Conv2D)          (None, 4, 4, 128)         131200    
_________________________________________________________________
max_pooling2d_49 (MaxPooling (None, 2, 2, 128)         0         
_________________________________________________________________
conv2d_107 (Conv2D)          (None, 1, 1, 64)          32832     
_________________________________________________________________
flatten_6 (Flatten)          (None, 64)                0         
_________________________________________________________________
dense_6 (Dense)              (None, 10)                650       
=================================================================
Total params: 1,378,570
Trainable params: 1,378,570
Non-trainable params: 0
_________________________________________________________________
```

In [69]:

```
history_3_a = model_3_a.fit(X_train, y_train, validation_data=(X_test, y_test), verbose=1, batch_si
ze=batch_size, epochs=epochs)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 20s 332us/step - loss: 0.1824 - acc: 0.9403 - val_l
oss: 0.0370 - val_acc: 0.9869
Epoch 2/15
60000/60000 [==============================] - 18s 292us/step - loss: 0.0418 - acc: 0.9868 - val_l
oss: 0.0327 - val_acc: 0.9896
Epoch 3/15
60000/60000 [==============================] - 18s 298us/step - loss: 0.0307 - acc: 0.9904 - val_l
oss: 0.0282 - val_acc: 0.9905
Epoch 4/15
60000/60000 [==============================] - 18s 300us/step - loss: 0.0234 - acc: 0.9927 - val_l
oss: 0.0298 - val_acc: 0.9910
Epoch 5/15
60000/60000 [==============================] - 18s 292us/step - loss: 0.0224 - acc: 0.9928 - val_l
oss: 0.0245 - val_acc: 0.9923
Epoch 6/15
60000/60000 [==============================] - 17s 290us/step - loss: 0.0166 - acc: 0.9947 - val_l
oss: 0.0318 - val_acc: 0.9902
Epoch 7/15
60000/60000 [==============================] - 17s 288us/step - loss: 0.0140 - acc: 0.9955 - val_l
oss: 0.0239 - val_acc: 0.9927
Epoch 8/15
60000/60000 [==============================] - 18s 294us/step - loss: 0.0139 - acc: 0.9955 - val_l
oss: 0.0415 - val_acc: 0.9877
Epoch 9/15
60000/60000 [==============================] - 18s 297us/step - loss: 0.0124 - acc: 0.9960 - val_l
oss: 0.0278 - val_acc: 0.9919
Epoch 10/15
60000/60000 [==============================] - 17s 290us/step - loss: 0.0100 - acc: 0.9971 - val_l
```

```
oss: 0.0365 - val_acc: 0.9902
Epoch 11/15
60000/60000 [==============================] - 17s 290us/step - loss: 0.0124 - acc: 0.9956 - val_l
oss: 0.0249 - val_acc: 0.9928
Epoch 12/15
60000/60000 [==============================] - 17s 291us/step - loss: 0.0083 - acc: 0.9975 - val_l
oss: 0.0423 - val_acc: 0.9893
Epoch 13/15
60000/60000 [==============================] - 17s 290us/step - loss: 0.0084 - acc: 0.9973 - val_l
oss: 0.0422 - val_acc: 0.9884
Epoch 14/15
60000/60000 [==============================] - 17s 290us/step - loss: 0.0077 - acc: 0.9976 - val_l
oss: 0.0321 - val_acc: 0.9915
Epoch 15/15
60000/60000 [==============================] - 17s 290us/step - loss: 0.0078 - acc: 0.9974 - val_l
oss: 0.0345 - val_acc: 0.9921
```

In [70]:

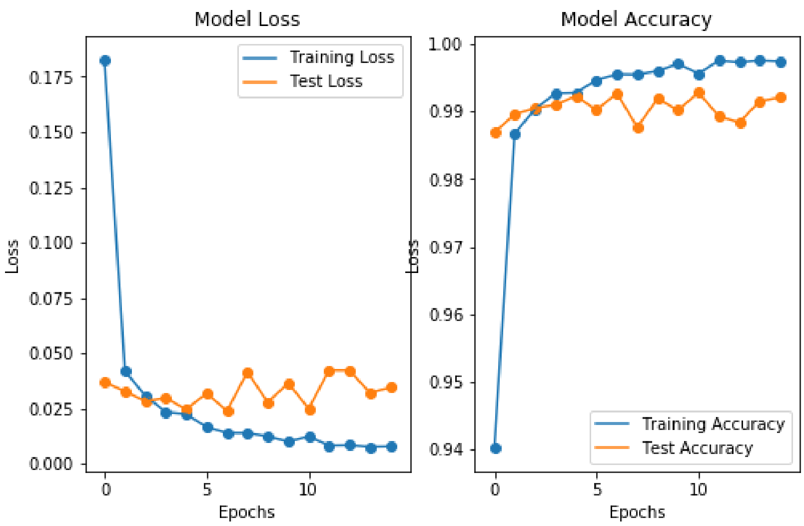
```python
score_3_a = model_3_a.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy without Dropout and Batch Norm', score_3_a[1])
```

```
Test Accuracy without Dropout and Batch Norm 0.9921
```

3.1.1 Epoch vs Loss , Epoch vs Accuracy

In [71]:

```python
fig = plt.figure(figsize=(8,5))

plt.subplot(1,2,1)
plt.plot(history_3_a.history['loss'], label='Training Loss')
plt.plot(history_3_a.history['val_loss'], label='Test Loss')
plt.scatter([i for i in range(epochs)],history_3_a.history['loss'] )
plt.scatter([i for i in range(epochs)],history_3_a.history['val_loss'] )
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history_3_a.history['acc'], label='Training Accuracy')
plt.plot(history_3_a.history['val_acc'], label='Test Accuracy')
plt.scatter([i for i in range(epochs)],history_3_a.history['acc'] )
plt.scatter([i for i in range(epochs)],history_3_a.history['val_acc'] )
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



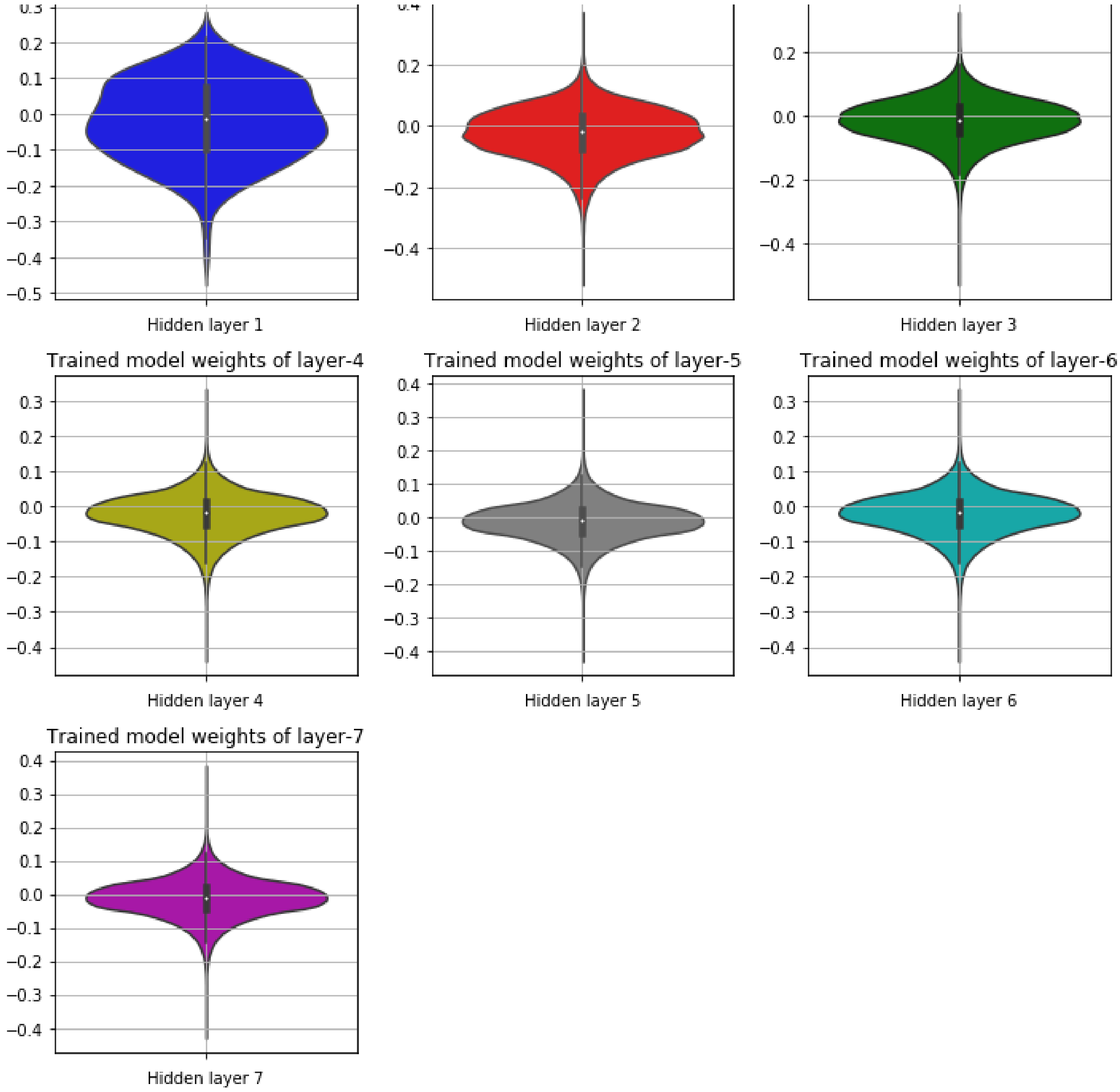
3.1.2 Distribution of Weights

In [0]:

```
w_after_3_a = model_3_a.get_weights()
h1_w = w_after_3_a[0].flatten().reshape(-1,1)
h2_w = w_after_3_a[2].flatten().reshape(-1,1)
h3_w = w_after_3_a[4].flatten().reshape(-1,1)
h4_w = w_after_3_a[6].flatten().reshape(-1,1)
h5_w = w_after_3_a[8].flatten().reshape(-1,1)
h6_w = w_after_3_a[10].flatten().reshape(-1,1)
h7_w = w_after_3_a[12].flatten().reshape(-1,1)
```

In [74]:

```
fig = plt.figure(figsize=(10, 10))
plt.title('Weights of the model after trained')

plt.subplot(3,3,1)
plt.title('Trained model weights of layer-1')
sns.violinplot(y=h1_w, color='b')
plt.xlabel('Hidden layer 1')
plt.grid()
plt.tight_layout()

plt.subplot(3,3,2)
plt.title('Trained model weights of layer-2')
sns.violinplot(y=h2_w, color='r')
plt.xlabel('Hidden layer 2')
plt.tight_layout()
plt.grid()


plt.subplot(3,3,3)
plt.title('Trained model weights of layer-3')
sns.violinplot(y=h3_w, color='g')
plt.xlabel('Hidden layer 3')
plt.grid()
plt.tight_layout()

plt.subplot(3,3,4)
plt.title('Trained model weights of layer-4')
sns.violinplot(y=h4_w, color='y')
plt.xlabel('Hidden layer 4')
plt.tight_layout()
plt.grid()


plt.subplot(3,3,5)
plt.title('Trained model weights of layer-5')
sns.violinplot(y=h5_w, color='gray')
plt.xlabel('Hidden layer 5')
plt.grid()
plt.tight_layout()

plt.subplot(3,3,6)
plt.title('Trained model weights of layer-6')
sns.violinplot(y=h4_w, color='c')
plt.xlabel('Hidden layer 6')
plt.tight_layout()
plt.grid()


plt.subplot(3,3,7)
plt.title('Trained model weights of layer-7')
sns.violinplot(y=h5_w, color='m')
plt.xlabel('Hidden layer 7')
plt.grid()
plt.tight_layout()

plt.show()
```



3.2 With Dropout and Batch Normalization

In [76]:

```
model_3_b = Sequential()

#conv layer 1
model_3_b.add(Conv2D(64, kernel_size=(2,2), activation='relu', input_shape=input_shape))
model_3_b.add(Dropout(0.2))

#conv layer 2
model_3_b.add(Conv2D(128, kernel_size=(2,2), activation='relu' ))
model_3_b.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model_3_b.add(Dropout(0.3))

#conv layer 3
model_3_b.add(Conv2D(256, kernel_size=(2,2), activation='relu'))
model_3_b.add(Dropout(0.4))


#conv layer 4
model_3_b.add(Conv2D(512, kernel_size=(2,2), activation='relu'))
model_3_b.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model_3_b.add(Dropout(0.5))


#conv layer 5
model_3_b.add(Conv2D(256, kernel_size=(2,2), activation='relu'))
model_3_b.add(Dropout(0.4))


#conv layer 6
model_3_b.add(Conv2D(128, kernel_size=(2,2), activation='relu'))
model_3_b.add(MaxPooling2D(pool_size=(2,2), padding='same'))
```

```
model_3_b.add(Dropout(0.3))


#conv layer 7
model_3_b.add(Conv2D(64, kernel_size=(2,2), activation='relu'))
model_3_b.add(Dropout(0.2))


#Flatten
model_3_b.add(Flatten())

#output class
model_3_b.add(Dense(num_classes, activation='softmax'))

#compile with adam optimizers
model_3_b.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#summary
model_3_b.summary()
```

```
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_113 (Conv2D)          (None, 27, 27, 64)        320
_________________________________________________________________
dropout_14 (Dropout)         (None, 27, 27, 64)        0
_________________________________________________________________
conv2d_114 (Conv2D)          (None, 26, 26, 128)       32896
_________________________________________________________________
max_pooling2d_54 (MaxPooling (None, 13, 13, 128)       0
_________________________________________________________________
dropout_15 (Dropout)         (None, 13, 13, 128)       0
_________________________________________________________________
conv2d_115 (Conv2D)          (None, 12, 12, 256)       131328
_________________________________________________________________
dropout_16 (Dropout)         (None, 12, 12, 256)       0
_________________________________________________________________
conv2d_116 (Conv2D)          (None, 11, 11, 512)       524800
_________________________________________________________________
max_pooling2d_55 (MaxPooling (None, 6, 6, 512)         0
_________________________________________________________________
dropout_17 (Dropout)         (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_117 (Conv2D)          (None, 5, 5, 256)         524544
_________________________________________________________________
dropout_18 (Dropout)         (None, 5, 5, 256)         0
_________________________________________________________________
conv2d_118 (Conv2D)          (None, 4, 4, 128)         131200
_________________________________________________________________
max_pooling2d_56 (MaxPooling (None, 2, 2, 128)         0
_________________________________________________________________
dropout_19 (Dropout)         (None, 2, 2, 128)         0
_________________________________________________________________
conv2d_119 (Conv2D)          (None, 1, 1, 64)          32832
_________________________________________________________________
dropout_20 (Dropout)         (None, 1, 1, 64)          0
_________________________________________________________________
flatten_7 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                650
=================================================================
Total params: 1,378,570
Trainable params: 1,378,570
Non-trainable params: 0
_________________________________________________________________
```

In [77]:

```
history_3_b = model_3_b.fit(X_train, y_train, validation_data=(X_test, y_test), verbose=1, batch_si
ze=batch_size, epochs=epochs)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 22s 359us/step - loss: 0.4536 - acc: 0.8506 - val_l
```

```
oss: 0.0577 - val_acc: 0.9833
Epoch 2/15
60000/60000 [==============================] - 21s 346us/step - loss: 0.1101 - acc: 0.9668 - val_l
oss: 0.0389 - val_acc: 0.9873
Epoch 3/15
60000/60000 [==============================] - 21s 346us/step - loss: 0.0796 - acc: 0.9759 - val_l
oss: 0.0313 - val_acc: 0.9906
Epoch 4/15
60000/60000 [==============================] - 20s 334us/step - loss: 0.0651 - acc: 0.9803 - val_l
oss: 0.0289 - val_acc: 0.9912
Epoch 5/15
60000/60000 [==============================] - 20s 333us/step - loss: 0.0609 - acc: 0.9817 - val_l
oss: 0.0248 - val_acc: 0.9925
Epoch 6/15
60000/60000 [==============================] - 20s 334us/step - loss: 0.0546 - acc: 0.9830 - val_l
oss: 0.0254 - val_acc: 0.9920
Epoch 7/15
60000/60000 [==============================] - 20s 335us/step - loss: 0.0512 - acc: 0.9847 - val_l
oss: 0.0211 - val_acc: 0.9939
Epoch 8/15
60000/60000 [==============================] - 20s 333us/step - loss: 0.0453 - acc: 0.9864 - val_l
oss: 0.0215 - val_acc: 0.9925
Epoch 9/15
60000/60000 [==============================] - 20s 332us/step - loss: 0.0445 - acc: 0.9864 - val_l
oss: 0.0191 - val_acc: 0.9933
Epoch 10/15
60000/60000 [==============================] - 20s 331us/step - loss: 0.0432 - acc: 0.9872 - val_l
oss: 0.0232 - val_acc: 0.9931
Epoch 11/15
60000/60000 [==============================] - 20s 332us/step - loss: 0.0421 - acc: 0.9877 - val_l
oss: 0.0202 - val_acc: 0.9935
Epoch 12/15
60000/60000 [==============================] - 20s 332us/step - loss: 0.0375 - acc: 0.9887 - val_l
oss: 0.0219 - val_acc: 0.9935
Epoch 13/15
60000/60000 [==============================] - 20s 332us/step - loss: 0.0371 - acc: 0.9889 - val_l
oss: 0.0209 - val_acc: 0.9935
Epoch 14/15
60000/60000 [==============================] - 20s 331us/step - loss: 0.0372 - acc: 0.9891 - val_l
oss: 0.0209 - val_acc: 0.9941
Epoch 15/15
60000/60000 [==============================] - 20s 332us/step - loss: 0.0352 - acc: 0.9895 - val_l
oss: 0.0169 - val_acc: 0.9945
```

In [78]:

```
score_3_b = model_3_b.evaluate(X_test, y_test, verbose=0)
print('Test accuracy with Dropout', score_3_b[1])
```

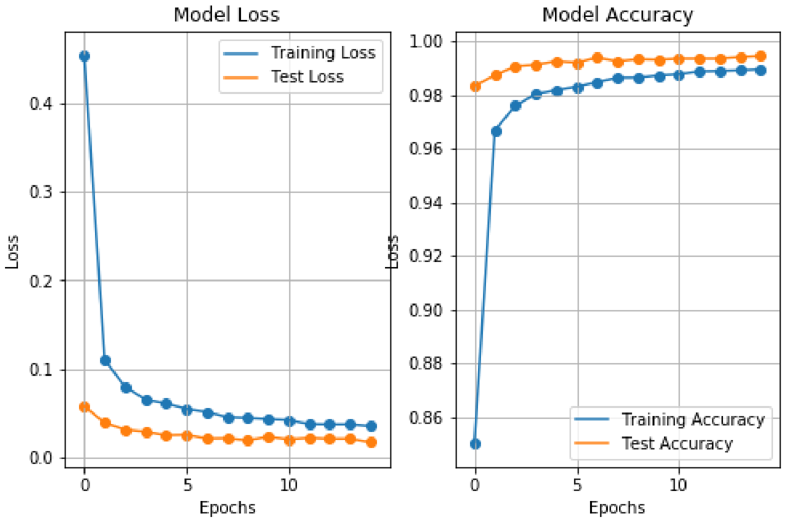
```
Test accuracy with BatchNorm and Dropout 0.9945
```

3.2.1 Loss vs Epoch , Epoch vs Accuracy

In [81]:

```
fig = plt.figure(figsize=(8,5))

plt.subplot(1,2,1)
plt.plot(history_3_b.history['loss'], label='Training Loss')
plt.plot(history_3_b.history['val_loss'], label='Test Loss')
plt.scatter([i for i in range(epochs)], history_3_b.history['loss'])
plt.scatter([i for i in range(epochs)], history_3_b.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()

plt.subplot(1,2,2)
plt.plot(history_3_b.history['acc'], label='Training Accuracy')
plt.plot(history_3_b.history['val_acc'], label='Test Accuracy')
plt.scatter([i for i in range(epochs)], history_3_b.history['acc'])
plt.scatter([i for i in range(epochs)], history_3_b.history['val_acc'])
```

```
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()

plt.show()
```



3.2.2 Distribution of Weights

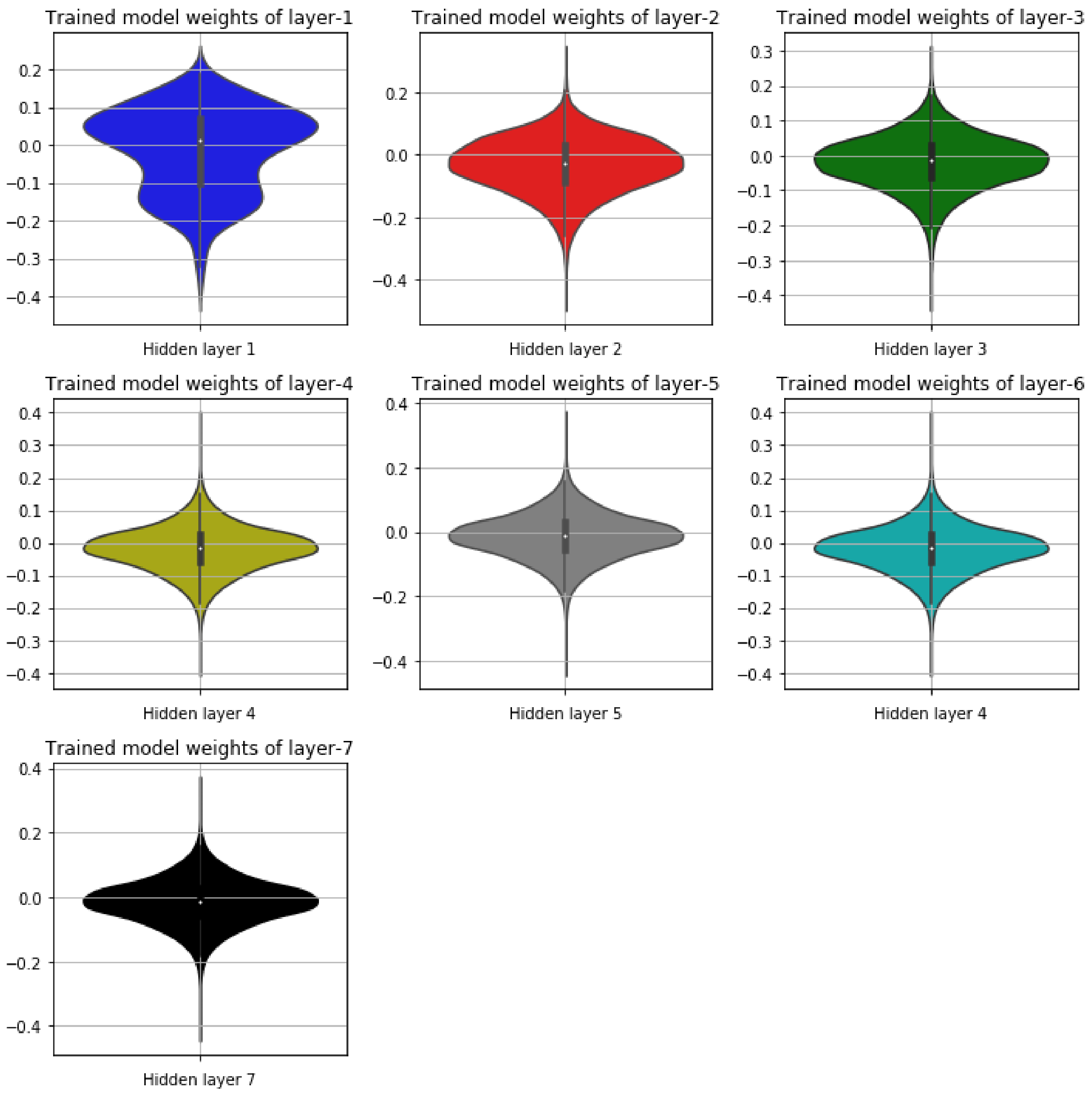
In [0]:

```
w_after_3_b = model_3_b.get_weights()
h1_w = w_after_3_b[0].flatten().reshape(-1,1)
h2_w = w_after_3_b[2].flatten().reshape(-1,1)
h3_w = w_after_3_b[4].flatten().reshape(-1,1)
h4_w = w_after_3_b[6].flatten().reshape(-1,1)
h5_w = w_after_3_b[8].flatten().reshape(-1,1)
h6_w = w_after_3_b[10].flatten().reshape(-1,1)
h7_w = w_after_3_b[12].flatten().reshape(-1,1)
```

In [85]:

```
fig = plt.figure(figsize=(10, 10))

plt.subplot(3,3,1)
plt.title('Trained model weights of layer-1')
sns.violinplot(y=h1_w, color='b')
plt.xlabel('Hidden layer 1')
plt.grid()
plt.tight_layout()

plt.subplot(3,3,2)
plt.title('Trained model weights of layer-2')
sns.violinplot(y=h2_w, color='r')
plt.xlabel('Hidden layer 2')
plt.tight_layout()
plt.grid()


plt.subplot(3,3,3)
plt.title('Trained model weights of layer-3')
sns.violinplot(y=h3_w, color='g')
plt.xlabel('Hidden layer 3')
plt.grid()
plt.tight_layout()

plt.subplot(3,3,4)
plt.title('Trained model weights of layer-4')
sns.violinplot(y=h4_w, color='y')
plt.xlabel('Hidden layer 4')
plt.tight_layout()
plt.grid()
```

```
plt.subplot(3,3,5)
plt.title('Trained model weights of layer-5')
sns.violinplot(y=h5_w, color='gray')
plt.xlabel('Hidden layer 5')
plt.grid()
plt.tight_layout()

plt.subplot(3,3,6)
plt.title('Trained model weights of layer-6')
sns.violinplot(y=h4_w, color='c')
plt.xlabel('Hidden layer 4')
plt.tight_layout()
plt.grid()


plt.subplot(3,3,7)
plt.title('Trained model weights of layer-7')
sns.violinplot(y=h5_w, color='k')
plt.xlabel('Hidden layer 7')
plt.grid()
plt.tight_layout()

plt.show()
```



Comparison

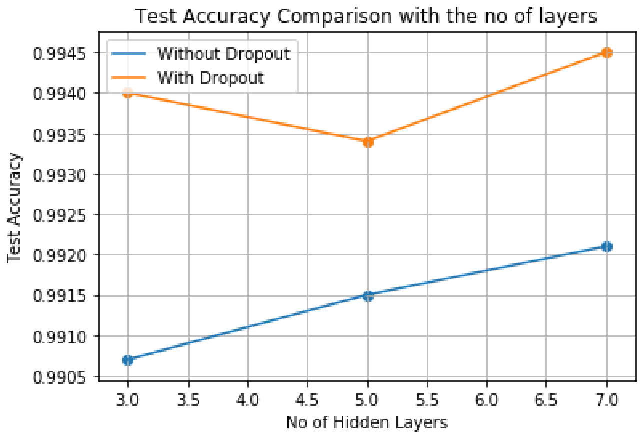
In [91]:

```
Test_accuracy_without_dropout = [score[1], score_2_a[1], score_3_a[1]]
Test_accuracy_with_dropout = [score_1_b[1], score_2_b[1], score_3_b[1]]
```

```python
no_of_hidden_layer = [3, 5, 7]

plt.plot(no_of_hidden_layer, Test_accuracy_without_dropout, label='Without Dropout')
plt.plot(no_of_hidden_layer, Test_accuracy_with_dropout, label='With Dropout')
plt.scatter(no_of_hidden_layer, Test_accuracy_without_dropout)
plt.scatter(no_of_hidden_layer, Test_accuracy_with_dropout)

plt.title('Test Accuracy Comparison with the no of layers')
plt.grid()
plt.xlabel('No of Hidden Layers')
plt.ylabel('Test Accuracy')
#plt.yscale('symlog')
plt.legend()
plt.plot()
```

Out[91]:

```
[]
```



Summary:

```
- Here also we can see that the Accuracy is high when we used Dropout
```

That's the end of the code