Traffic Sign Detection

[1] Business Problem

Traffic signs are an important part of road safety when it comes to driving vehicles. They provide critical information using which users can avoid road accidents. These are enforced by the government to ensure road safety.

The objective here is to automatically classify a traffic sign using deep learning techniques.

Source - http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset (http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset)

[2] About the data

- This is a single image multiclass classification problem.
- Dataset contains a total of 43 classes.
- More than 50,000 images (data points).

[2.1] Image format

- The images contain one traffic sign each.
- Images are stored in PPM(portable pixmap) format.
- Image sizes vary between 15x15 to 250x250 pixels.
- Images are not necessarily squared.

[2.2] Annotations format

Annotations are provided in CSV files. Fields are separated by ";" (semicolon). Annotations contain the following information:

- **Filename:** Filename of corresponding image.
- **Width:** Width of the image.
- **Height:** Height of the image.
- **ROI.x1:** X-coordinate of top-left corner of traffic sign bounding box.
- **ROI.y1:** Y-coordinate of top-left corner of traffic sign bounding box.
- **ROI.x2:** X-coordinate of bottom-right corner of traffic sign bounding box.
- **ROI.y2:** Y-coordinate of bottom-right corner of traffic sign bounding box.
- **ClassId:** Assigned class label.

[3] Data Munging

[3.1] Data download and extract

In [0]:

```
!ls
```

```
data  GTSRB_Final_Training_Images.zip  images  sample_data
```

In [0]:

```
!wget https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/GTSRB_Final_Trai
!mkdir images
!unzip GTSRB_Final_Training_Images.zip -d images
!ls
```

```
--2019-06-21 04:11:42--  https://sid.erda.dk/public/archives/daaeac0d7ce11
52aea9b61d9f1e19370/GTSRB_Final_Training_Images.zip (https://sid.erda.dk/p
ublic/archives/daaeac0d7ce1152aea9b61d9f1e19370/GTSRB_Final_Training_Image
s.zip)
Resolving sid.erda.dk (sid.erda.dk)... 130.225.104.13
Connecting to sid.erda.dk (sid.erda.dk)|130.225.104.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 276294756 (263M) [application/zip]
Saving to: ‘GTSRB_Final_Training_Images.zip.1’

GTSRB_Final_Trainin 100%[===================>] 263.50M  81.8MB/s    in 3.2
s

2019-06-21 04:11:46 (81.8 MB/s) - ‘GTSRB_Final_Training_Images.zip.1’ save
d [276294756/276294756]

mkdir: cannot create directory ‘images’: File exists
Archive:  GTSRB_Final_Training_Images.zip
  inflating: images/GTSRB/Final_Training/Images/00000/00000_00000.ppm
```

In [0]:

```
!ls images/GTSRB/Final_Training/Images
```

```
00000  00004  00008  00012  00016  00020  00024  00028  00032  00036  00040
00001  00005  00009  00013  00017  00021  00025  00029  00033  00037  00041
00002  00006  00010  00014  00018  00022  00026  00030  00034  00038  00042
00003  00007  00011  00015  00019  00023  00027  00031  00035  00039
```

In [0]:

```
#importing necessary libraries
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import os
import glob
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from tqdm import tqdm
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import f1_score

import keras
from keras.models import Model
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from keras.layers.normalization import BatchNormalization

dir_path = 'images/GTSRB/Final_Training/Images'
```

```
Using TensorFlow backend.
```

In [0]:

```
# each class has a separate directory and each directory contains a csv
# so here we are loading all the csv and merging them into one dataframe while
# we preserve the classes
flag = False
for dir_ in os.listdir(dir_path):
    if flag == False:
        df = pd.read_csv(os.path.join(dir_path, dir_, 'GT-'+dir_+'.csv'), sep=';')
        flag = True
    else:
        df_temp = pd.read_csv(os.path.join(dir_path, dir_, 'GT-'+dir_+'.csv'), sep=';')
        df = pd.concat([df, df_temp])
df.reset_index(drop=True, inplace=True)
df.head()
```

Out[14]:

	Filename	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId
0	00000_00000.ppm	25	25	5	5	20	20	13
1	00000_00001.ppm	26	27	5	6	21	22	13
2	00000_00002.ppm	26	28	5	6	21	22	13
3	00000_00003.ppm	28	28	5	5	23	23	13
4	00000_00004.ppm	29	29	5	5	23	23	13

In [0]:

```
# moving the images into one folder for better accessibility
!mkdir data
for dir_ in os.listdir(dir_path):
    for img in os.listdir(os.path.join(dir_path, dir_)):
        if img.endswith('.ppm'):
            os.rename(os.path.join(dir_path, dir_, img), os.path.join('data', str(int(dir_)
!ls data
```

```
mkdir: cannot create directory ‘data’: File exists
0_00000_00000.ppm   13_00064_00023.ppm  25_00037_00016.ppm  38_00066_0001
0.ppm
0_00000_00001.ppm   13_00064_00024.ppm  25_00037_00017.ppm  38_00066_0001
1.ppm
0_00000_00002.ppm   13_00064_00025.ppm  25_00037_00018.ppm  38_00066_0001
2.ppm
0_00000_00003.ppm   13_00064_00026.ppm  25_00037_00019.ppm  38_00066_0001
3.ppm
0_00000_00004.ppm   13_00064_00027.ppm  25_00037_00020.ppm  38_00066_0001
4.ppm
0_00000_00005.ppm   13_00064_00028.ppm  25_00037_00021.ppm  38_00066_0001
5.ppm
0_00000_00006.ppm   13_00064_00029.ppm  25_00037_00022.ppm  38_00066_0001
6.ppm
0_00000_00007.ppm   13_00065_00000.ppm  25_00037_00023.ppm  38_00066_0001
7.ppm
0_00000_00008.ppm   13_00065_00001.ppm  25_00037_00024.ppm  38_00066_0001
8.ppm
```

In [0]:

```
df['Filename'] = df['ClassId'].astype(str) + '_' + df['Filename']
df.head()
```

Out[16]:

	Filename	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId
0	13_00000_00000.ppm	25	25	5	5	20	20	13
1	13_00000_00001.ppm	26	27	5	6	21	22	13
2	13_00000_00002.ppm	26	28	5	6	21	22	13
3	13_00000_00003.ppm	28	28	5	5	23	23	13
4	13_00000_00004.ppm	29	29	5	5	23	23	13

In [0]:

```
df['ClassId'].value_counts()
```

Out[7]:

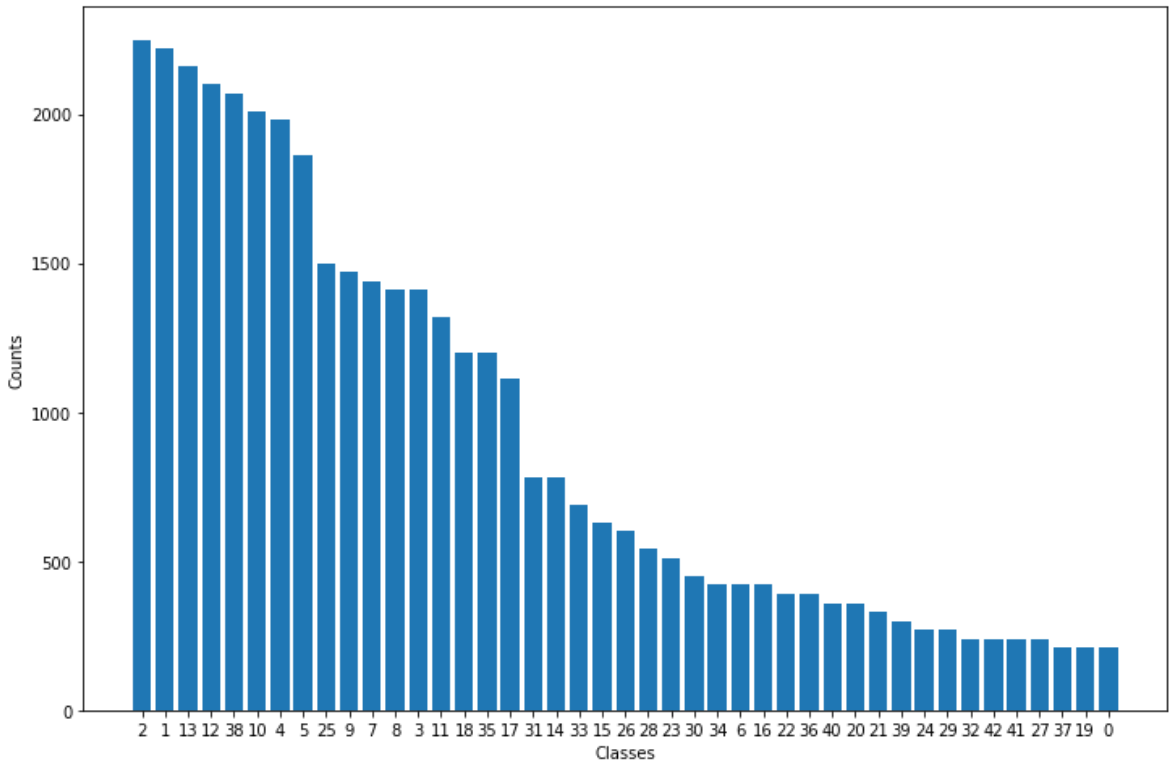
```
2     2250
1     2220
13    2160
12    2100
38    2070
10    2010
4     1980
5     1860
25    1500
9     1470
7     1440
8     1410
3     1410
11    1320
18    1200
35    1200
17    1110
31     780
14     780
33     689
15     630
26     600
28     540
23     510
30     450
34     420
6      420
16     420
22     390
36     390
40     360
20     360
21     330
39     300
24     270
29     270
32     240
42     240
41     240
27     240
37     210
19     210
0      210
Name: ClassId, dtype: int64
```

[3.2] Distribution of classes

We have an im-balanced data here, with most of data points belonging to class 2 and the least of the data points belonging to class 0.

In [0]:

```
plt.figure(figsize=(12, 8))
x = df['ClassId'].value_counts().index
y = df['ClassId'].value_counts().values
plt.bar([str(i) for i in x], y)
plt.xlabel("Classes")
plt.ylabel("Counts")
plt.show()
```



[3.3] Image cropping and Grey scale conversion

- We have been given the bounding box positions for each which contains the traffic sign. We'll use those coordinates to extract the traffic sign for our classification problem for better accuracy.
- We find the optimal image size for training so that while we reshape each image, losing information is avoided for most of the images.
- We convert each image into Grey scale for training our network.

In [0]:

```
im = Image.open('data/27_00000_00000.ppm')
im
```

Out[9]:

In [0]:

```
df['Width'].describe()
```

Out[10]:

```
count    39209.000000
mean        50.835880
std         24.306933
min         25.000000
25%         35.000000
50%         43.000000
75%         58.000000
max        243.000000
Name: Width, dtype: float64
```

In [0]:

```
df['Height'].describe()
```

Out[11]:

```
count    39209.000000
mean        50.328930
std         23.115423
min         25.000000
25%         35.000000
50%         43.000000
75%         58.000000
max        225.000000
Name: Height, dtype: float64
```

In [0]:

```
im.size
```

Out[12]:

```
(43, 41)
```

In [0]:

```
im_new = im.resize((64, 64))
im_new
```

Out[13]:

In [0]:

```
df.loc[df['Height'].idxmax()]
```

Out[14]:

```
Filename    25_00003_00029.ppm
Width                      243
Height                     225
Roi.X1                      20
Roi.Y1                      20
Roi.X2                     223
Roi.Y2                     205
ClassId                     25
Name: 18569, dtype: object
```

In [0]:

```
im_temp = Image.open('data/25_00003_00029.ppm')
im_temp = im_temp.crop((20, 20, 223, 205))
im_temp
```

Out[15]:

In [0]:

```
im_temp.resize((64, 64))
```

Out[16]:

In [0]:

```
# this code snippet contains code for cropping the images based on coordinates
# and resizing the image
!mkdir cropped_data
for idx, row in tqdm(df.iterrows(), total=df.shape[0]):
    im = Image.open('data/'+row['Filename'])
    im_crop = im.crop((row['Roi.X1'], row['Roi.Y1'], row['Roi.X2'], row['Roi.Y2']))
    im_resize = im_crop.resize((64, 64))
    im_resize.save('cropped_data/'+row['Filename'])
```

```
100%|██████████| 39209/39209 [00:18<00:00, 2139.92it/s]
```

In [0]:

```
# this code snippet contains code for grey scale conversion of each image
!mkdir grey_data
for idx, row in tqdm(df.iterrows(), total=df.shape[0]):
    im = Image.open('cropped_data/'+row['Filename']).convert('L')
    im.save('grey_data/'+row['Filename'].split(".")[0]+".png", format='png')
```

```
100%|██████████| 39209/39209 [00:25<00:00, 1517.46it/s]
```

In [0]:

```
# we extract features from each image in ndarray format
labels = []
img_feats_grey = []
for img in tqdm(os.listdir('grey_data')):
        im = Image.open('grey_data/'+img)
        img_feats_grey.append(np.array(im))
        labels.append(int(img.split("_")[0]))
img_feats_grey = np.array(img_feats_grey)
img_feats_grey.shape, len(labels)
```

```
100%|██████████| 39209/39209 [00:06<00:00, 6266.33it/s]
```

Out[19]:

```
((39209, 64, 64), 39209)
```

[3.4] Train - Test split

In [0]:

```
x_train, x_test, y_train, y_test = train_test_split(img_feats_grey, labels, test_size=0.3,
x_train.shape, len(y_train)
```

Out[20]:

```
((27446, 64, 64), 27446)
```

[3.5] Data normalization and reshaping

In [0]:

```
input_shape = (64, 64, 1)
```

In [0]:

```
batch_size = 128
num_classes = 43

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
x_train = x_train.reshape(x_train.shape[0], 64, 64, 1)
x_test = x_test.reshape(x_test.shape[0], 64, 64, 1)
x_train.shape, len(y_train)
```

Out[22]:

```
((27446, 64, 64, 1), 27446)
```

[4] Utility Functions

- **plt_dynamic:** A function that plots a graph given training and validation loss.
- **display_activation:** A function that displays the feature (visualization) of each kernel given an image, layer number and the number of visualizations you want in each row and column.
- **display_feature_spike:** We take the output of each kernel in a layer and take the mean of values of each kernel output. For example, if we have 64 kernels (3x3) in a 2nd layer and we input an image of shape 64x64, we get an output of 62x62x64. If we take the mean of all 62x62 values, then we have a total of 64 means which we plot in a graph in this function. If we get a spike at 12 index in the graph for example, then we know for this particular input image, 12th kernel in 2nd layer responds!
- **display_activation_kernel:** Once we get the kernel number from display_feature_spike() and layer number, we display the output image (after it passes through the kernel).
- **call_utilities_features:** Calls all the above functions one after the other. Also prints the probablities of top 5 classes in a bar graph.

In [0]:

```
# utility functions

def plt_dynamic(fig, x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()

def display_activation(activations, col_size, row_size, act_index):
    activation = activations[act_index]
    activation_index=0
    fig, ax = plt.subplots(row_size, col_size, figsize=(row_size*2.5,col_size*1.5))
    for row in range(0,row_size):
        for col in range(0,col_size):
            ax[row][col].imshow(activation[0, :, :, activation_index], cmap='gray')
            activation_index += 1
    plt.show()
    print("Multiple kernel activations starting from 0")

def display_activation_kernel(activations, col_size, row_size, act_index, kernel_index):
    activation = activations[act_index]
    plt.figure(figsize=(7, 4))
    plt.imshow(activation[0, :, :, kernel_index], cmap='gray')
    plt.show()
    print("Activation for", kernel_index, "kernel in", act_index, "layer.")

#https://towardsdatascience.com/how-to-visualize-convolutional-features-in-40-lines-of-code
def display_feature_spike(activations, col_size, row_size, act_index):
    activation = activations[act_index]
    means = []
    for act in range(activation.shape[3]):
        means.append(np.mean(activation[0, :, :, act]))
    print("The kernel which activates/recognizes the shape:", np.argmax(means))
    plt.figure(figsize=(12, 8))
    plt.plot([str(i) for i in range(len(means))], means)
    plt.xticks(rotation=90)
    plt.show()
    return np.argmax(means)
```

In [0]:

```
def call_utilities_features(img_idx, activations, col_size, row_size, act_index, y_true, y_
    print("Actual class it belongs to:", y_true[img_idx])
    print("Predicted class", np.argmax(y_pred[img_idx]))
    print("Actual training image")
    plt.imshow(x_test[img_idx].reshape(64,64))
    plt.show()
    kernel_index = display_feature_spike(activations, col_size, row_size, act_index)
    display_activation(activations, col_size, row_size, act_index)
    display_activation_kernel(activations, col_size, row_size, act_index, kernel_index)
    classes = np.array([str(i) for i in range(num_classes)])
    probs = y_pred[img_idx]
    idxs = probs.argsort()
    probs = probs[idxs][::-1][:5]
    classes = classes[idxs][::-1][:5]
    plt.figure()
    plt.bar(classes, probs)
    plt.show()
    print("Probabilities of top 5 classes.")
```

[5] Models building

[5.1] Conv(8 -- 3x3) - Conv(16 -- 3x3) - MaxPool(2x2) - Dropout(0.75) - Dense(128) - Dropout(0.5)

In [0]:

```
epochs = 10
model = Sequential()
model.add(Conv2D(
    8, kernel_size=(3, 3), activation='relu', input_shape=input_shape
))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.75))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(
    loss=keras.losses.categorical_crossentropy,
    optimizer=keras.optimizers.Adadelta(), metrics=['accuracy']
)

model.summary()

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 62, 62, 8)         80
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 60, 60, 16)        1168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 16)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 30, 30, 16)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 14400)             0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               1843328
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 43)                5547
=================================================================
Total params: 1,850,123
Trainable params: 1,850,123
Non-trainable params: 0
_________________________________________________________________
Train on 27446 samples, validate on 11763 samples
Epoch 1/10
27446/27446 [==============================] - 4s 136us/step - loss: 2.3137
- acc: 0.3871 - val_loss: 0.8539 - val_acc: 0.7669
Epoch 2/10
27446/27446 [==============================] - 3s 118us/step - loss: 0.9240
- acc: 0.7361 - val_loss: 0.3998 - val_acc: 0.9029
Epoch 3/10
27446/27446 [==============================] - 3s 118us/step - loss: 0.6322
- acc: 0.8201 - val_loss: 0.2831 - val_acc: 0.9375
Epoch 4/10
```

```
27446/27446 [==============================] - 3s 118us/step - loss: 0.5144
- acc: 0.8528 - val_loss: 0.2442 - val_acc: 0.9431
Epoch 5/10
27446/27446 [==============================] - 3s 119us/step - loss: 0.4418
- acc: 0.8717 - val_loss: 0.1877 - val_acc: 0.9565
Epoch 6/10
27446/27446 [==============================] - 3s 119us/step - loss: 0.3977
- acc: 0.8812 - val_loss: 0.1702 - val_acc: 0.9614
Epoch 7/10
27446/27446 [==============================] - 3s 118us/step - loss: 0.3547
- acc: 0.8967 - val_loss: 0.1531 - val_acc: 0.9653
Epoch 8/10
27446/27446 [==============================] - 3s 119us/step - loss: 0.3328
- acc: 0.9001 - val_loss: 0.1466 - val_acc: 0.9657
Epoch 9/10
27446/27446 [==============================] - 3s 118us/step - loss: 0.3099
- acc: 0.9095 - val_loss: 0.1408 - val_acc: 0.9700
Epoch 10/10
27446/27446 [==============================] - 3s 121us/step - loss: 0.2823
- acc: 0.9177 - val_loss: 0.1241 - val_acc: 0.9730
```

In [0]:

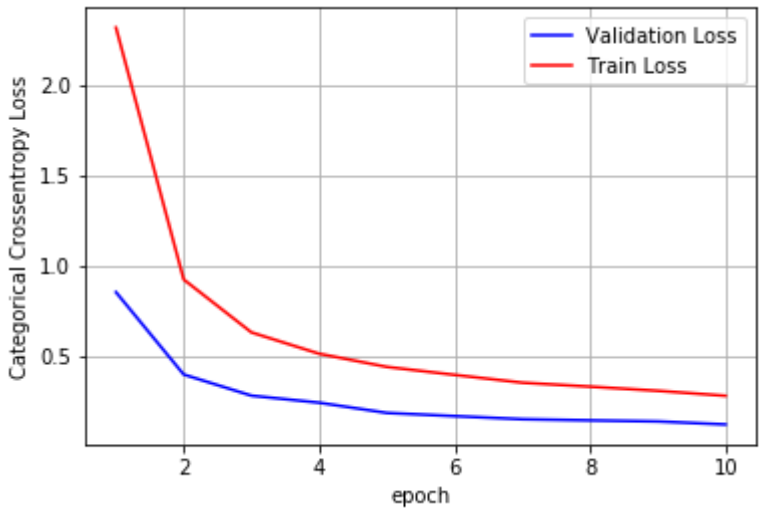
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(fig, x, vy, ty, ax)
```

```
Test score: 0.12410678104621399
Test accuracy: 0.9729660801170817
```



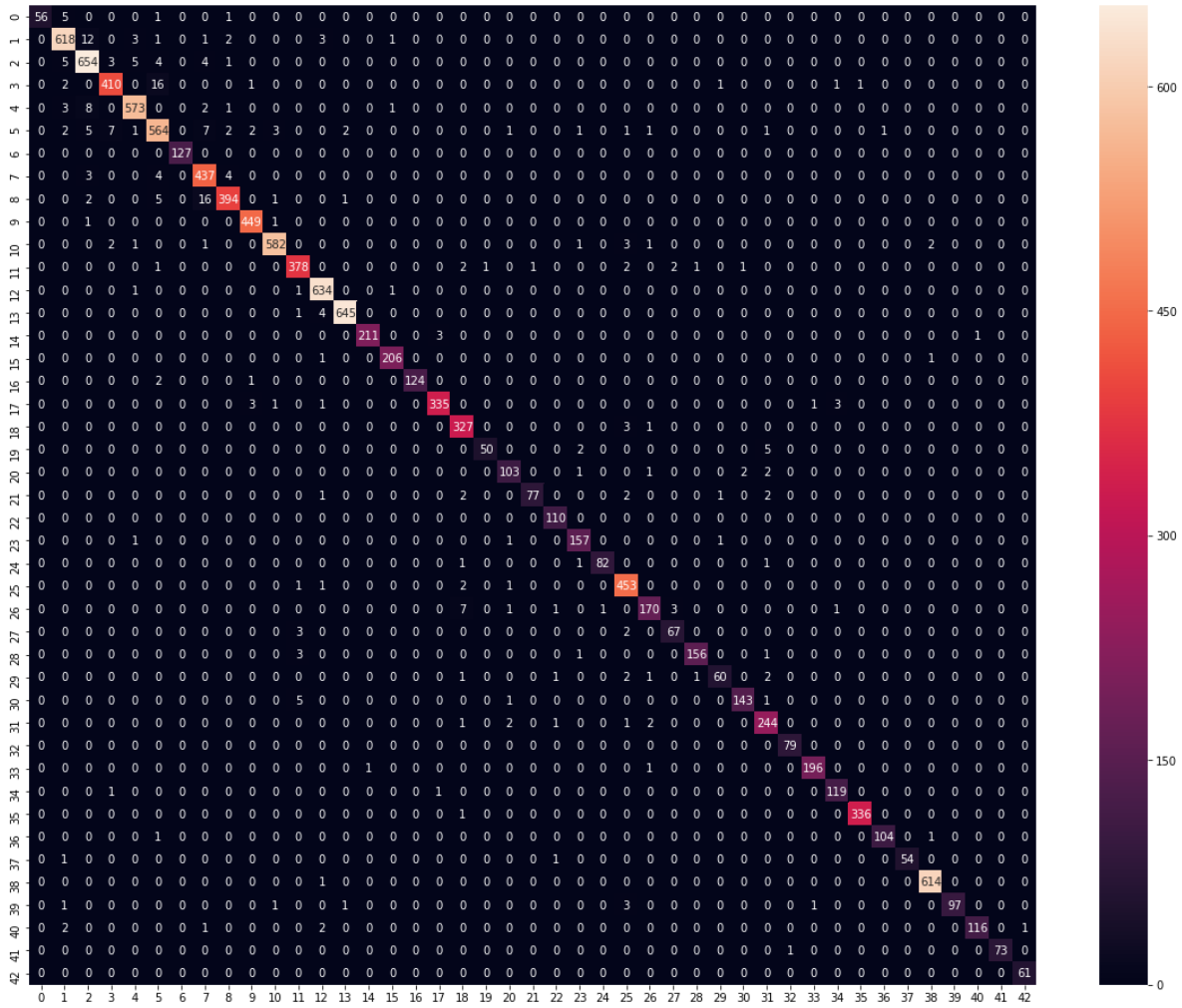
In [0]:

```
plt.figure(figsize=(20,16))
y_prediction = model.predict(x_test)
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_prediction, axis = 1)
# Convert validation observations to one hot vectors
y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
print("-----------------------------------------------------------")
print("Micro F1 score", f1_score(y_true, y_pred_classes, average='micro'))
print("-----------------------------------------------------------")
confusion_mtx = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(confusion_mtx, annot=True, fmt="d")
```

```
-----------------------------------------------------------
Micro F1 score 0.9729660800816118
-----------------------------------------------------------
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b9feb6240>
```



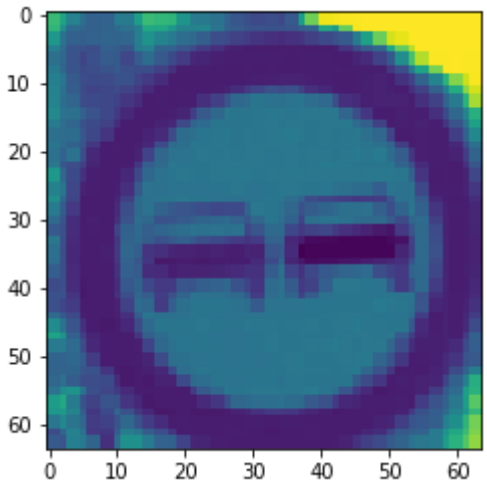
In [0]:

```
layer_outputs = [layer.output for layer in model.layers]
activation_model = Model(inputs=model.input, outputs=layer_outputs)
```

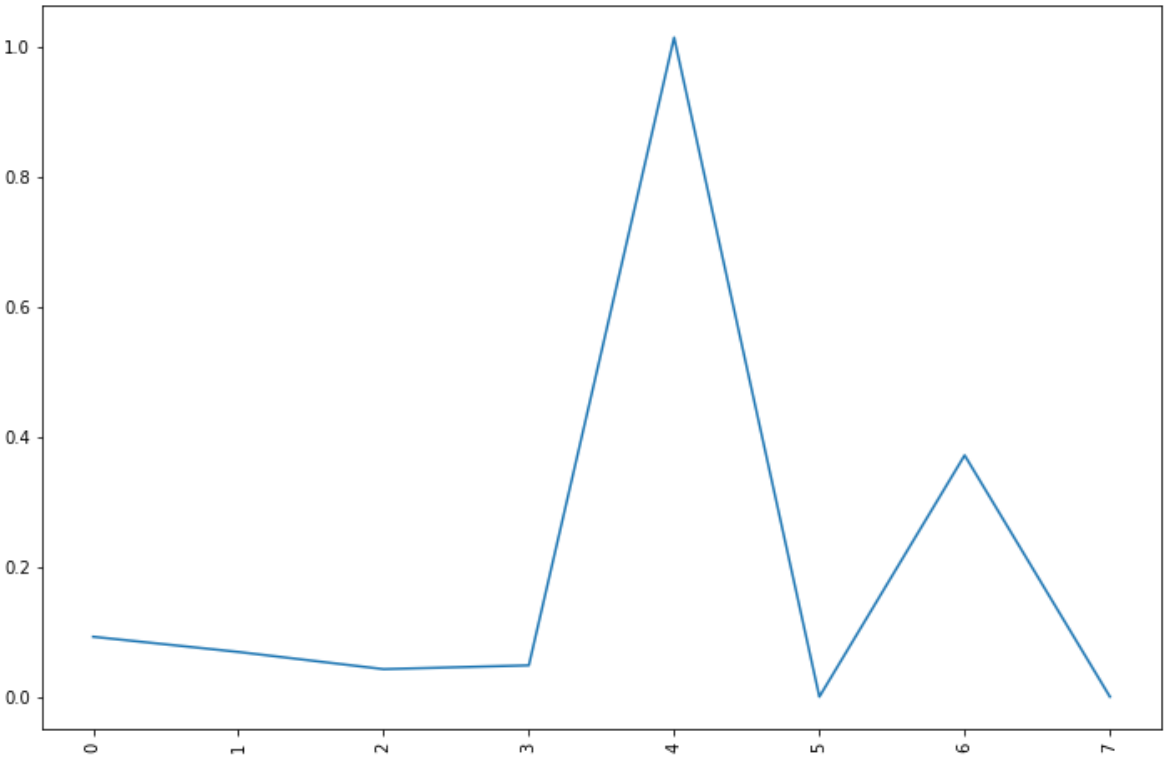
In [0]:

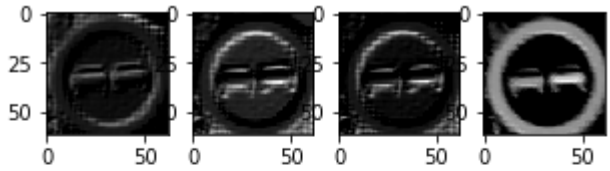
```
idx = 25
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 2, 0, y_true, y_prediction)
```

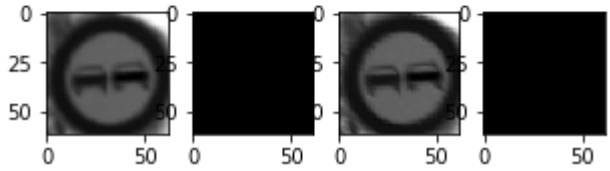
```
Actual class it belongs to: 9
Predicted class 9
Actual training image
```



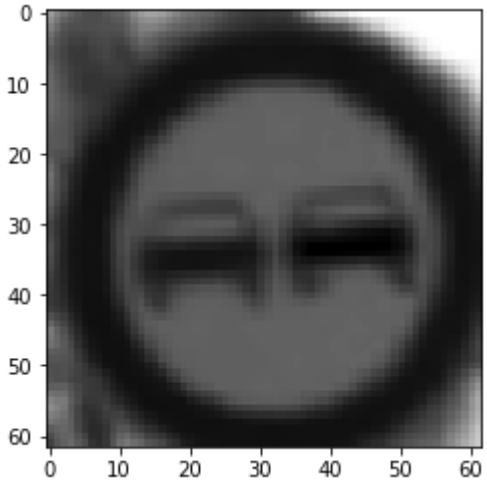

```
The kernel which activates/recognizes the shape: 4
```



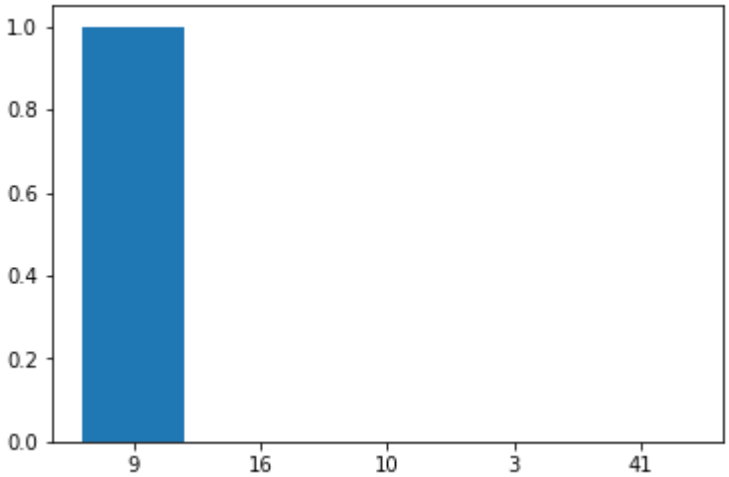




Multiple kernel activations starting from 0



Activation for 4 kernel in 0 layer.

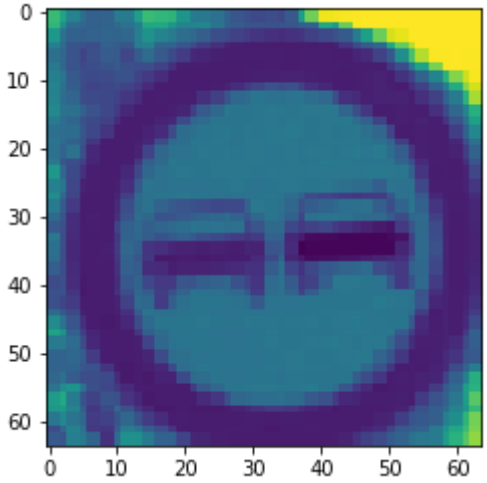


Probabilities of top 5 classes.

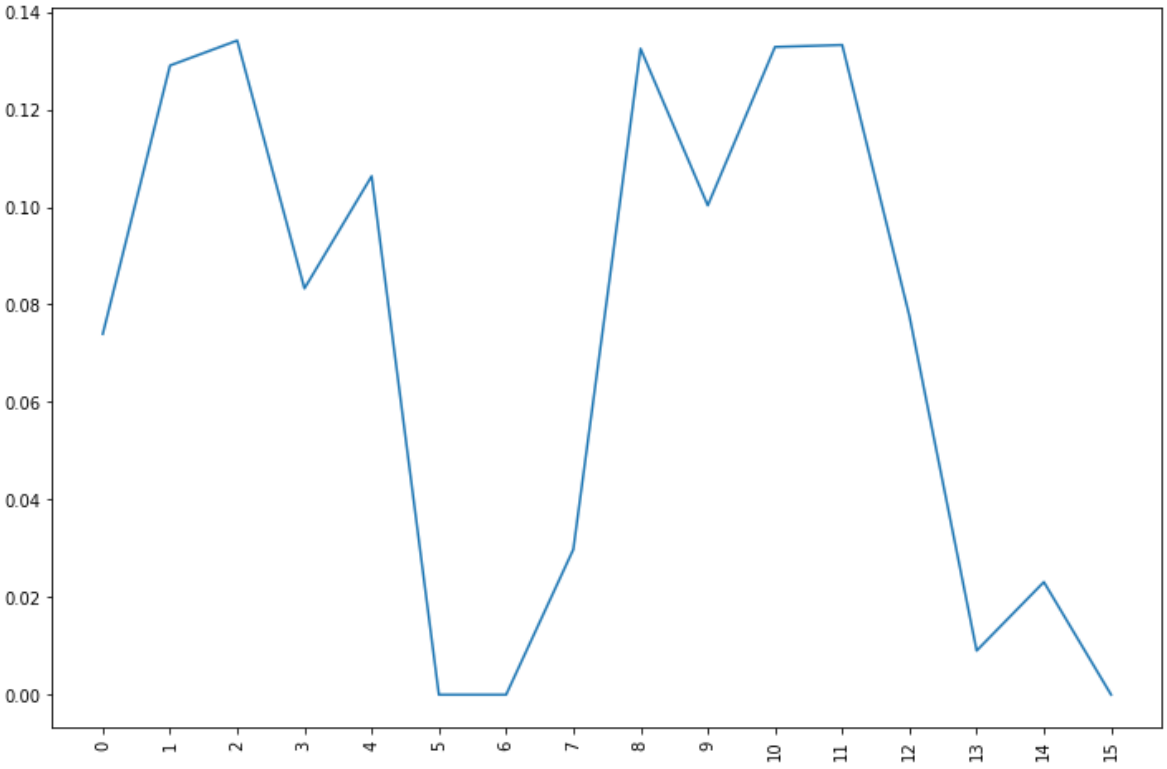
In [0]:

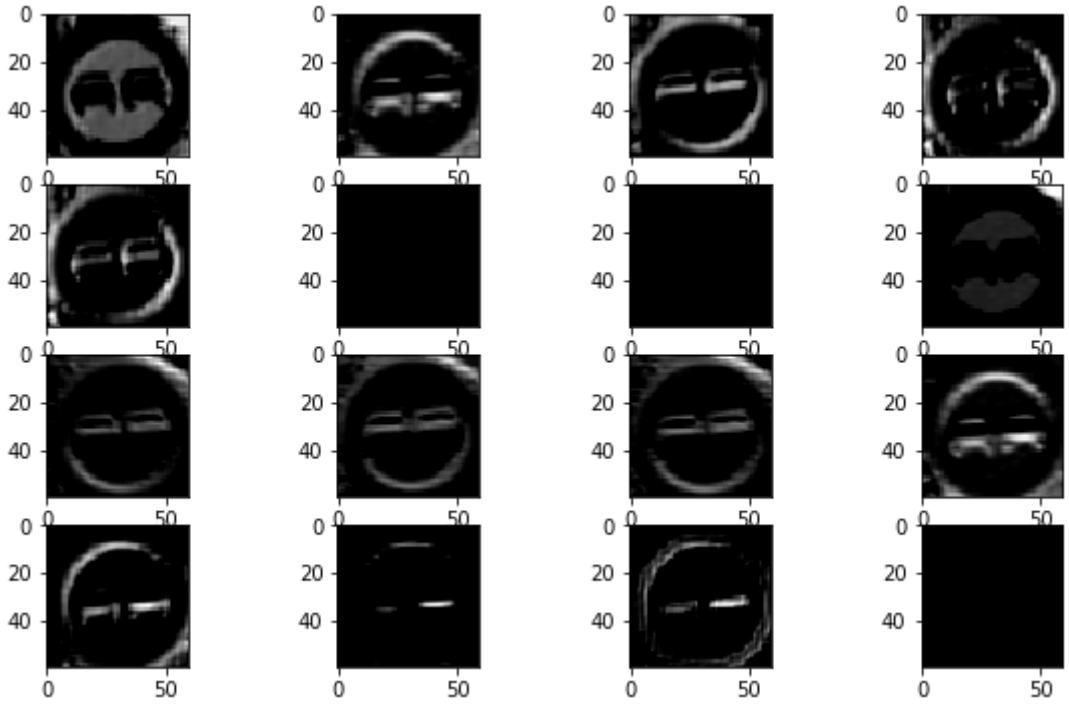
```
call_utilities_features(idx, activations, 4, 4, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 9
Predicted class 9
Actual training image
```

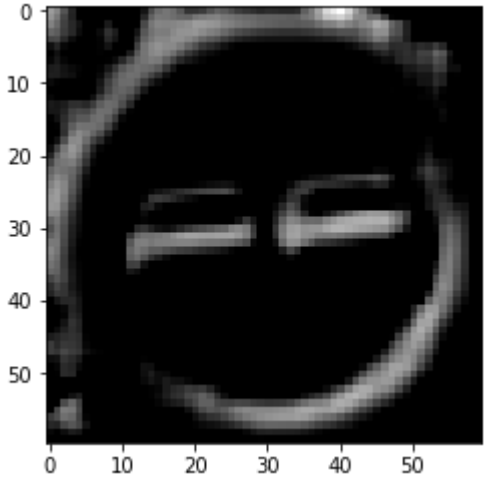



```
The kernel which activates/recognizes the shape: 2
```

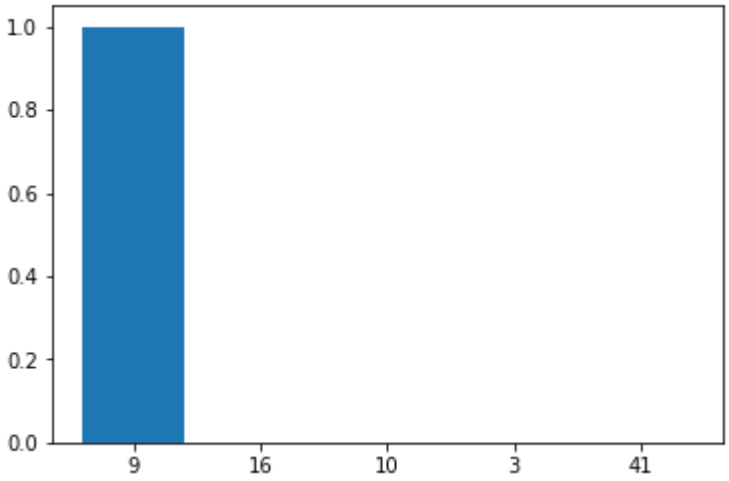




Multiple kernel activations starting from 0



Activation for 2 kernel in 1 layer.

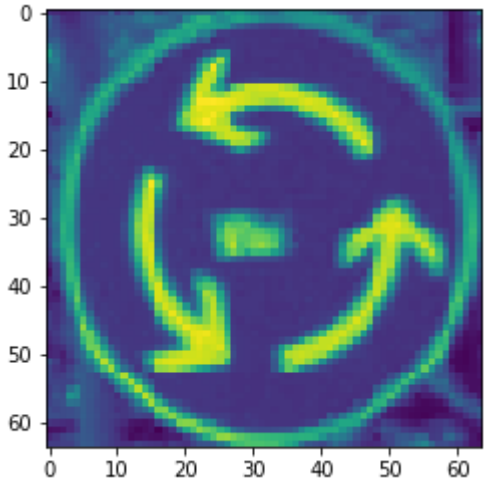


Probabilities of top 5 classes.

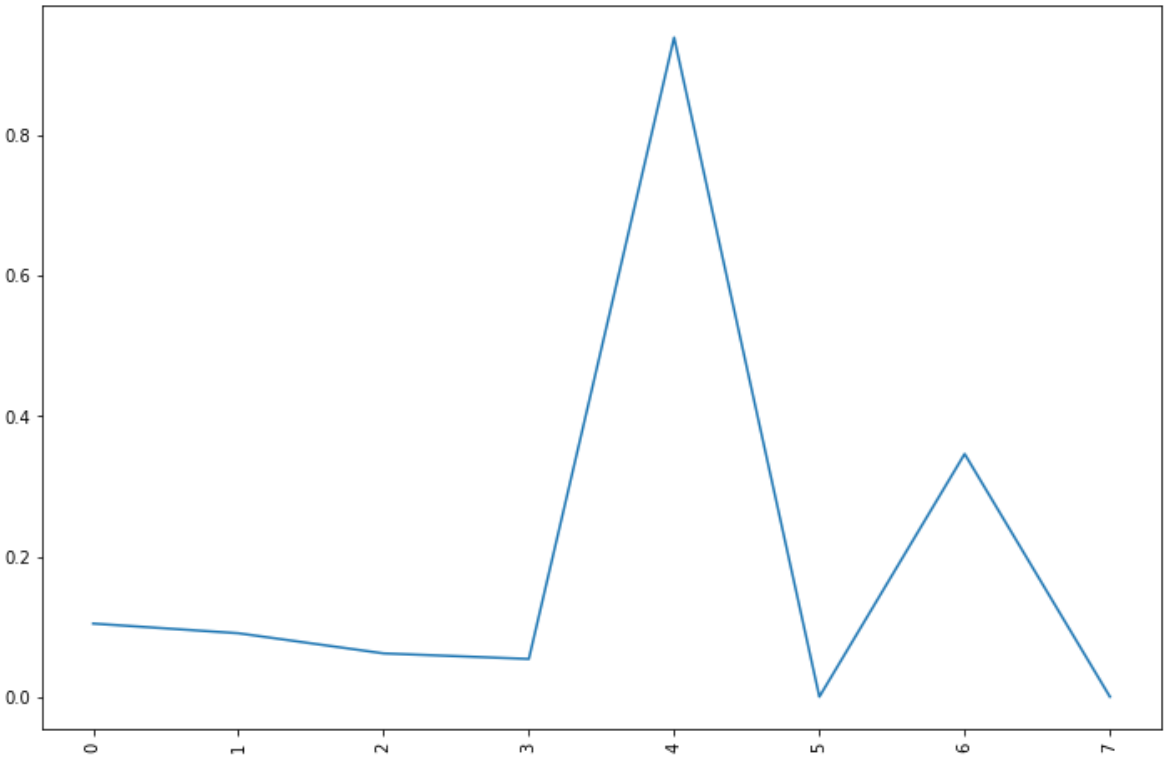
In [0]:

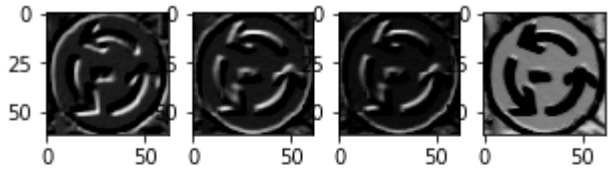
```
idx = 100
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 2, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 4, 4, 1, y_true, y_prediction)
```

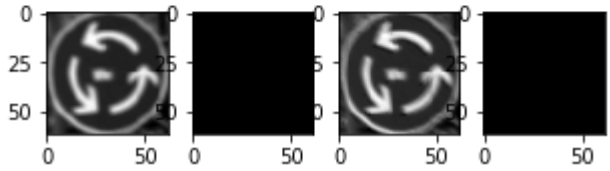
```
Actual class it belongs to: 40
Predicted class 40
Actual training image
```



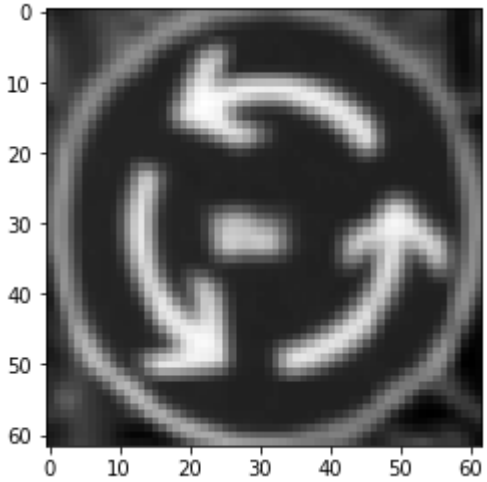

```
The kernel which activates/recognizes the shape: 4
```



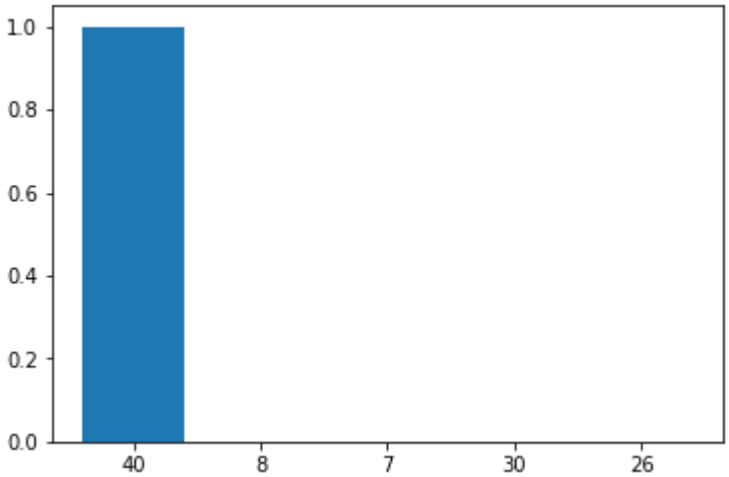




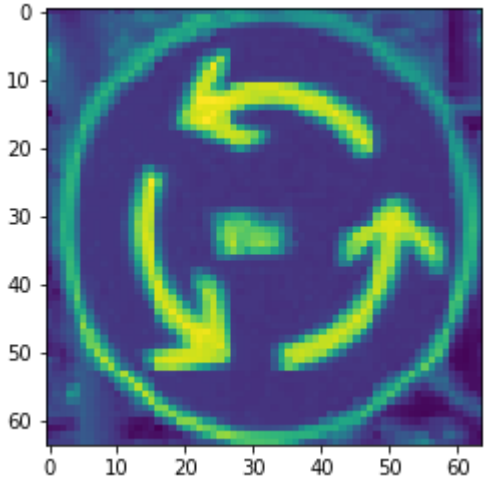
Multiple kernel activations starting from 0



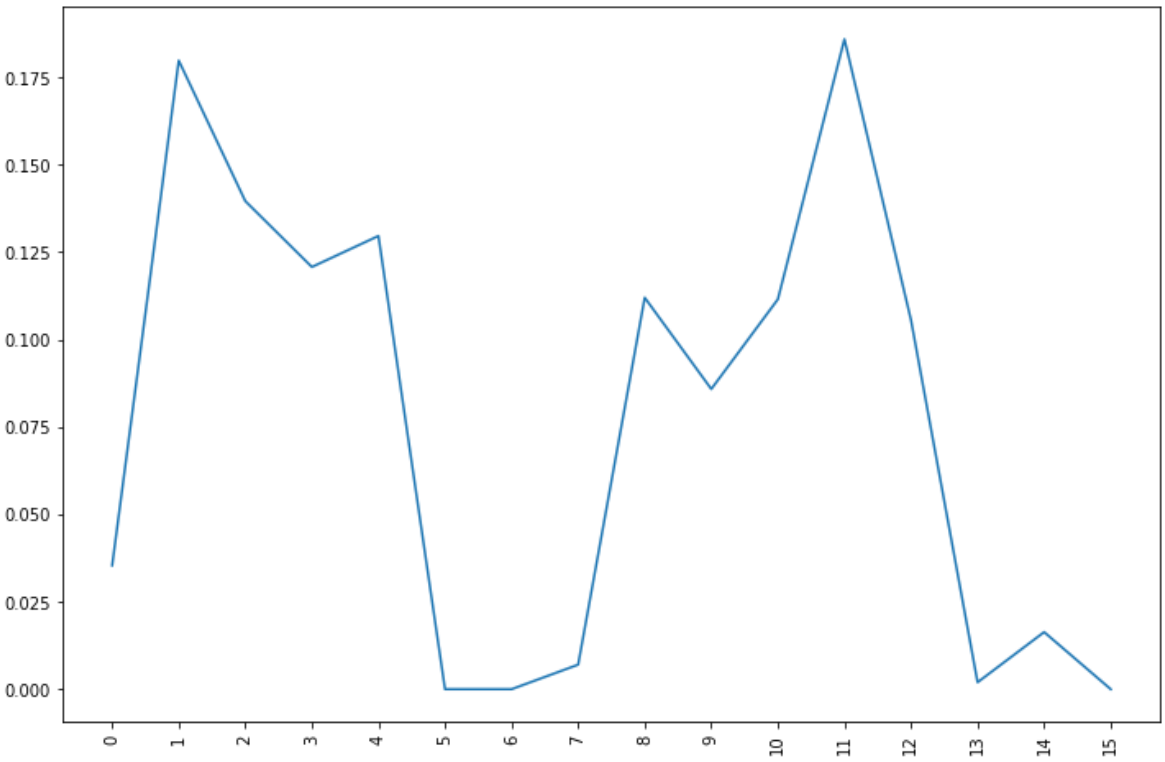
Activation for 4 kernel in 0 layer.

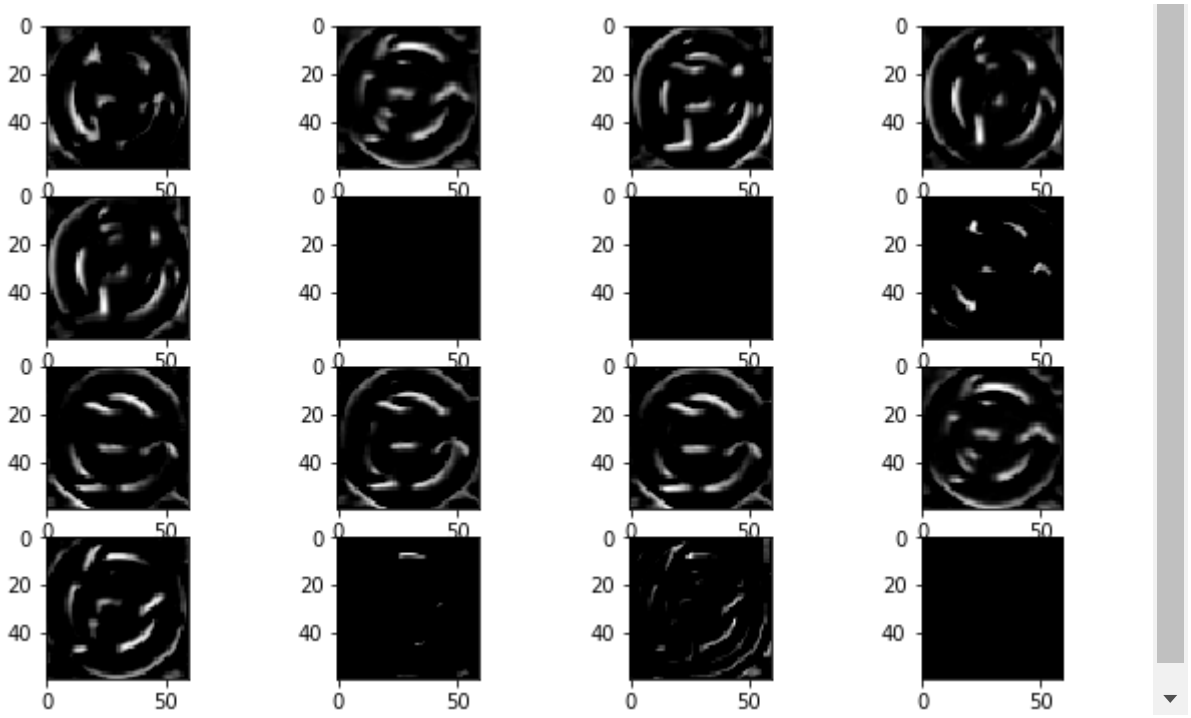


Probabilities of top 5 classes.
Actual class it belongs to: 40
Predicted class 40
Actual training image

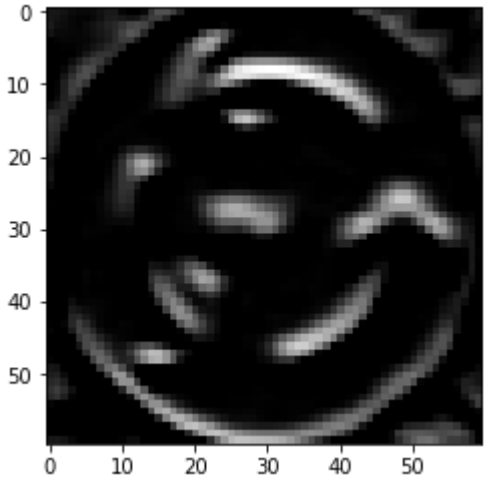



```
The kernel which activates/recognizes the shape: 11
```

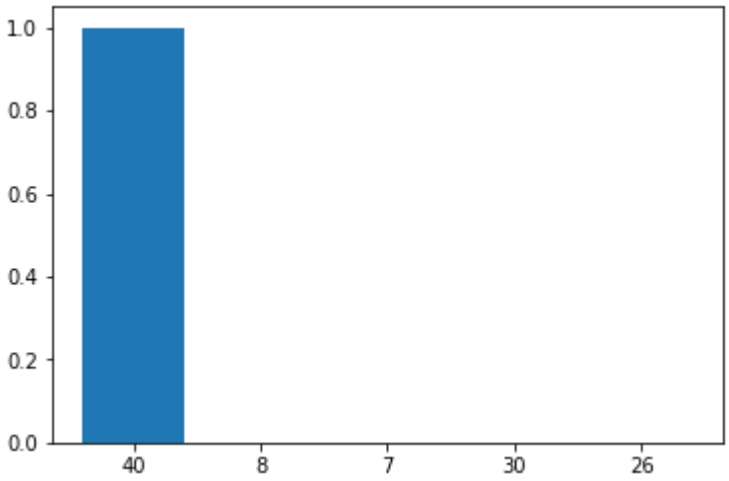




Multiple kernel activations starting from 0



Activation for 11 kernel in 1 layer.



Probabilities of top 5 classes.

In [0]:

```
actual_class = []
index_image = []
predicted_class = []
for idx, im in enumerate(y_true):
    if im != np.argmax(y_prediction[idx]):
        actual_class.append(im)
        predicted_class.append(np.argmax(y_prediction[idx]))
        index_image.append(idx)
np.array(index_image)
```

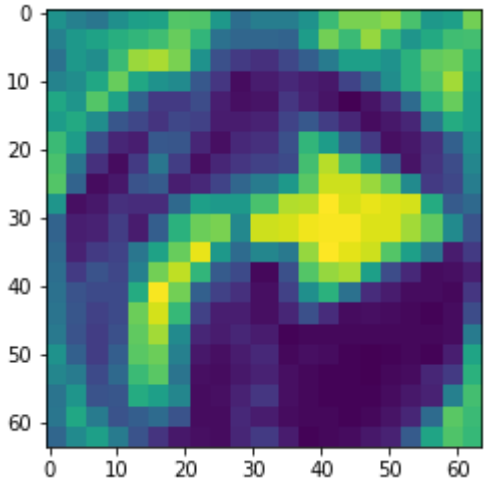
Out[42]:

```
array([   11,    20,    41,    88,   126,   151,   157,   163,   215,
         250,   258,   358,   365,   434,   569,   576,   592,   643,
         684,   696,   803,   836,   882,   981,  1053,  1097,  1105,
        1107,  1108,  1145,  1148,  1152,  1182,  1188,  1210,  1243,
        1275,  1278,  1314,  1320,  1321,  1391,  1394,  1478,  1485,
        1578,  1587,  1609,  1689,  1728,  1737,  1748,  1750,  1755,
        1774,  1811,  1840,  1880,  1915,  1926,  1998,  2028,  2033,
        2045,  2082,  2166,  2218,  2283,  2288,  2336,  2361,  2516,
        2534,  2657,  2680,  2722,  2735,  2791,  2831,  2869,  2873,
        2925,  2978,  3050,  3055,  3113,  3450,  3470,  3489,  3510,
        3513,  3523,  3530,  3551,  3576,  3592,  3665,  3669,  3730,
        3846,  3853,  3909,  3962,  3989,  4028,  4046,  4083,  4144,
        4157,  4277,  4302,  4316,  4319,  4369,  4375,  4435,  4477,
        4553,  4558,  4611,  4636,  4646,  4738,  4743,  4889,  4921,
        4926,  4953,  4986,  5001,  5010,  5032,  5037,  5050,  5051,
        5082,  5089,  5189,  5223,  5240,  5269,  5273,  5292,  5308,
        5341,  5417,  5446,  5491,  5513,  5618,  5671,  5722,  5802,
        5823,  5844,  5855,  5959,  6017,  6018,  6028,  6037,  6094,
        6102,  6122,  6128,  6157,  6165,  6166,  6167,  6175,  6209,
        6339,  6372,  6384,  6408,  6454,  6464,  6554,  6584,  6620,
        6669,  6790,  6795,  6805,  6806,  6894,  6985,  7003,  7072,
        7086,  7121,  7126,  7269,  7288,  7341,  7375,  7377,  7378,
        7436,  7481,  7496,  7521,  7570,  7582,  7626,  7637,  7678,
        7700,  7725,  7745,  7763,  7796,  7799,  7833,  7861,  7871,
        7883,  7914,  7931,  7941,  7954,  7980,  7990,  8047,  8080,
        8090,  8174,  8200,  8218,  8235,  8279,  8345,  8346,  8348,
        8412,  8440,  8492,  8510,  8525,  8542,  8578,  8598,  8627,
        8632,  8723,  8735,  8739,  8767,  8876,  8887,  9090,  9200,
        9314,  9326,  9518,  9584,  9608,  9740,  9749,  9800,  9840,
        9865,  9873,  9942,  9966, 10073, 10094, 10098, 10124, 10125,
       10220, 10241, 10302, 10306, 10370, 10467, 10498, 10554, 10569,
       10621, 10805, 10814, 10882, 10898, 10917, 10943, 10969, 10972,
       10990, 11018, 11042, 11063, 11075, 11078, 11090, 11120, 11144,
       11211, 11220, 11253, 11301, 11307, 11332, 11333, 11354, 11366,
       11400, 11404, 11432, 11483, 11527, 11601, 11653, 11684, 11697,
       11711, 11722, 11754])
```

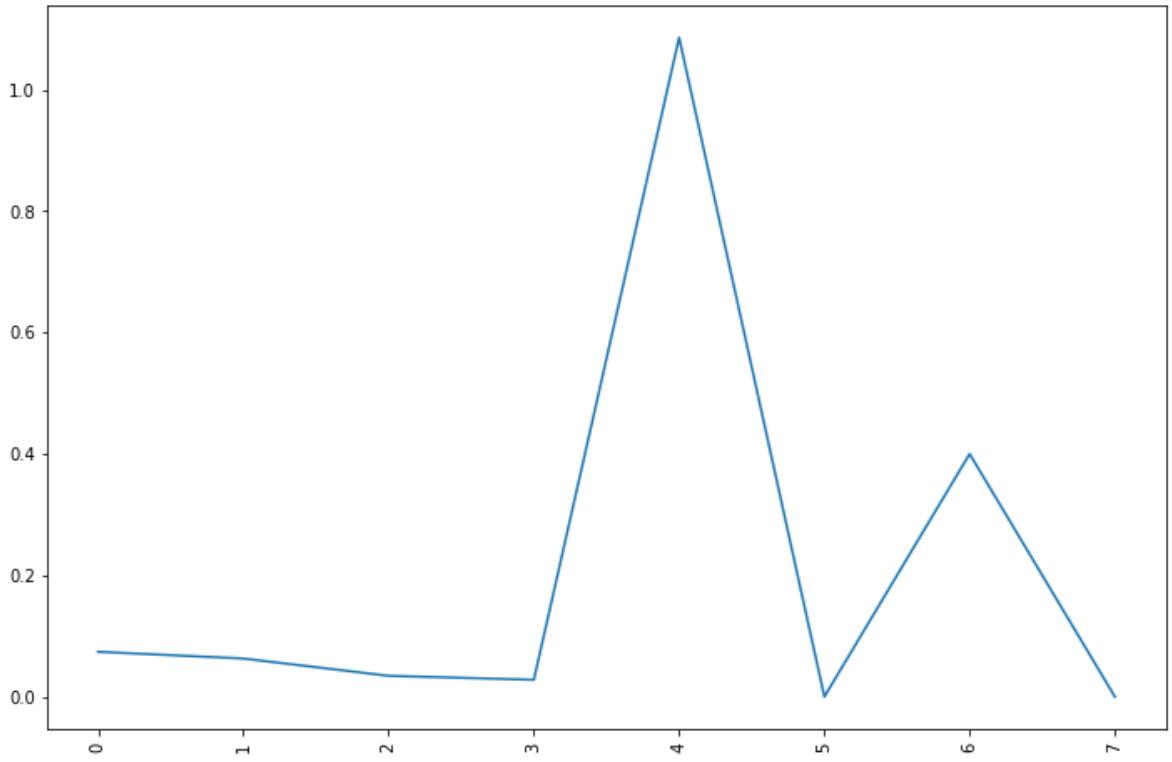
In [0]:

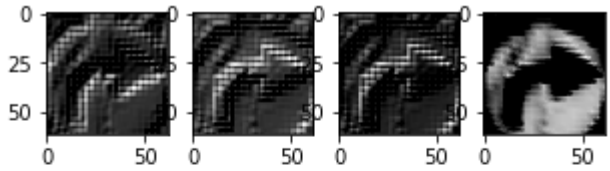
```
idx = 9942
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 2, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 4, 4, 1, y_true, y_prediction)
```

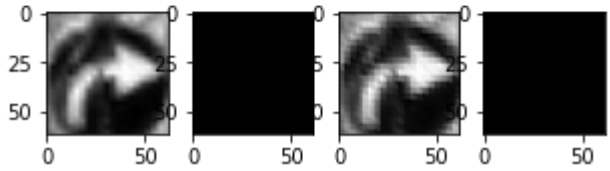
```
Actual class it belongs to: 33
Predicted class 14
Actual training image
```



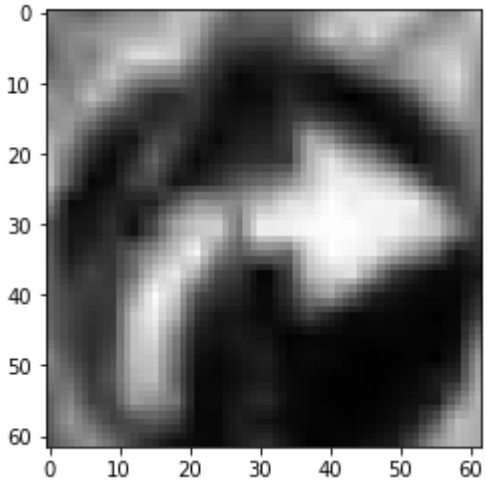

```
The kernel which activates/recognizes the shape: 4
```



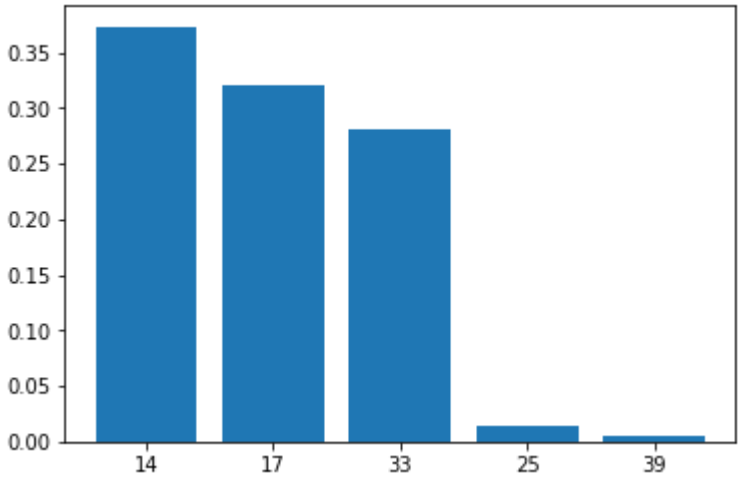




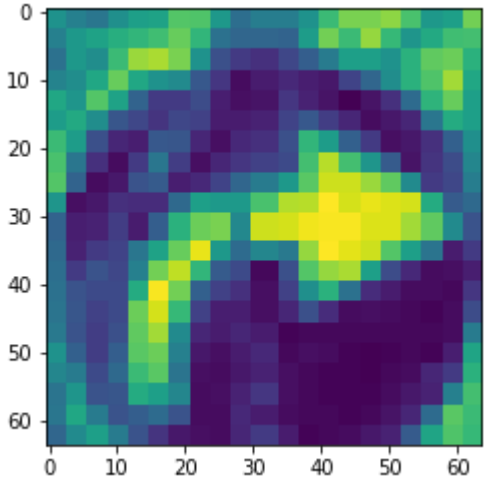
Multiple kernel activations starting from 0



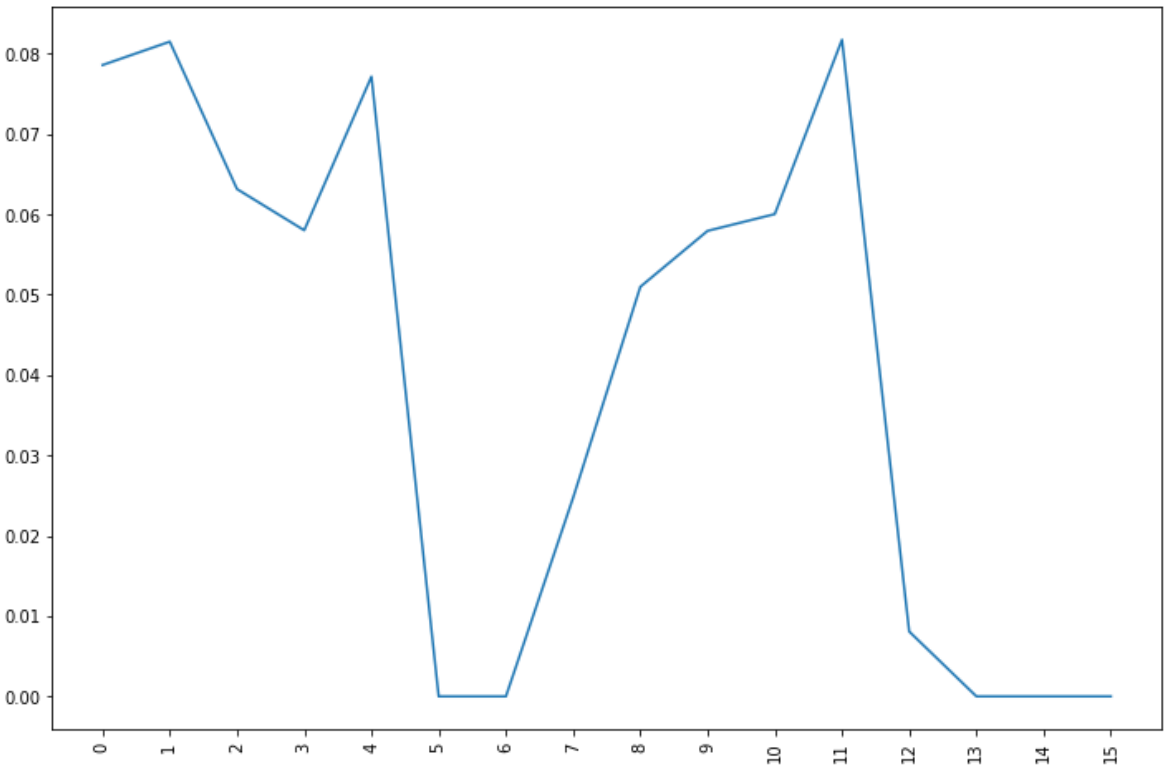
Activation for 4 kernel in 0 layer.

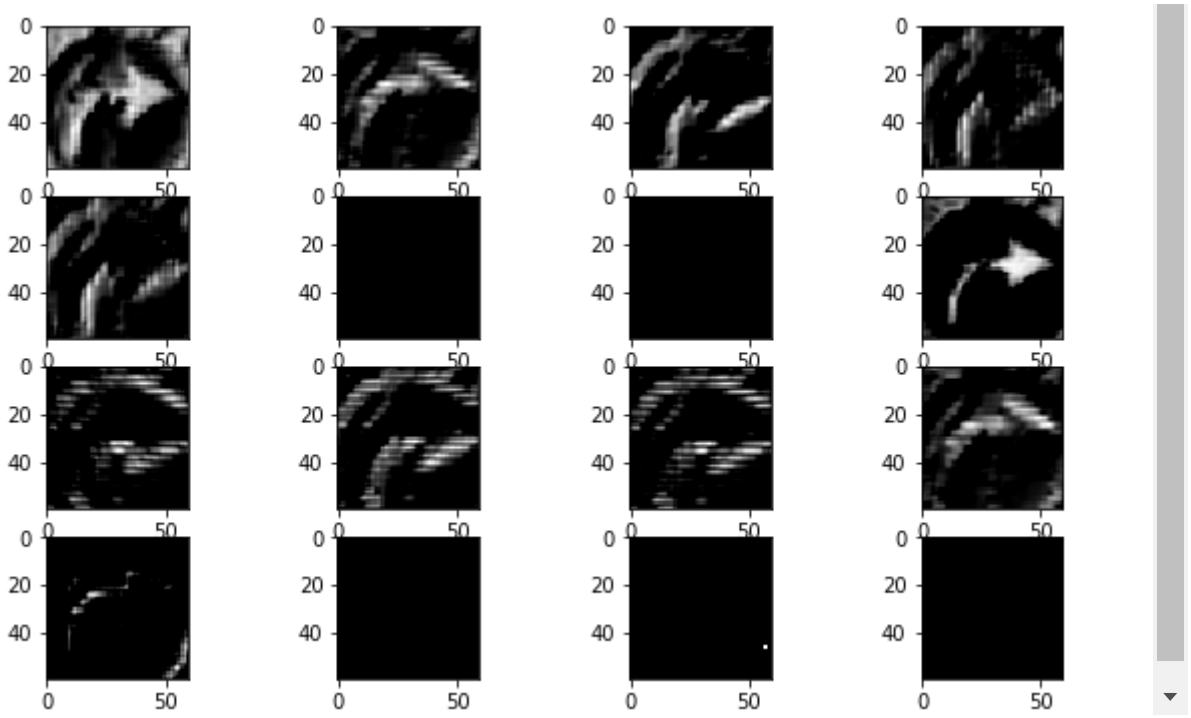


Probabilities of top 5 classes.
Actual class it belongs to: 33
Predicted class 14
Actual training image

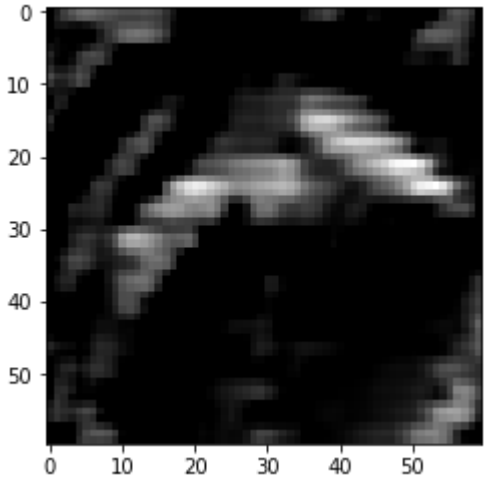



```
The kernel which activates/recognizes the shape: 11
```

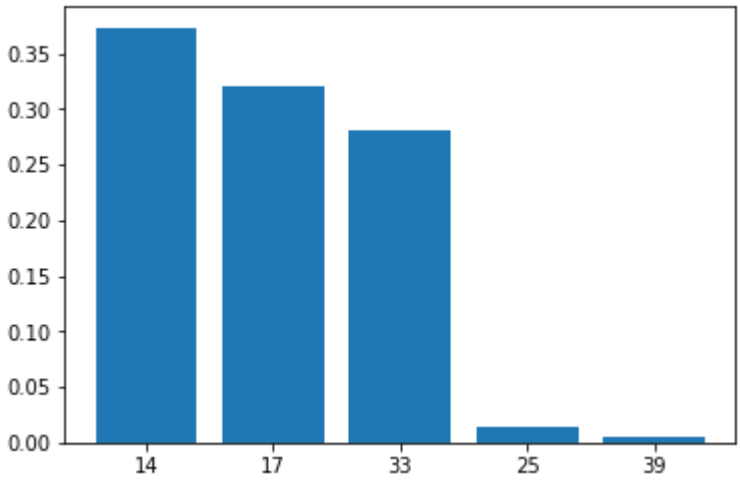




Multiple kernel activations starting from 0



Activation for 11 kernel in 1 layer.

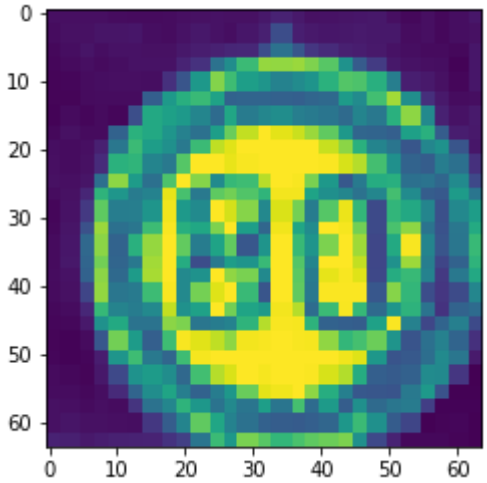


Probabilities of top 5 classes.

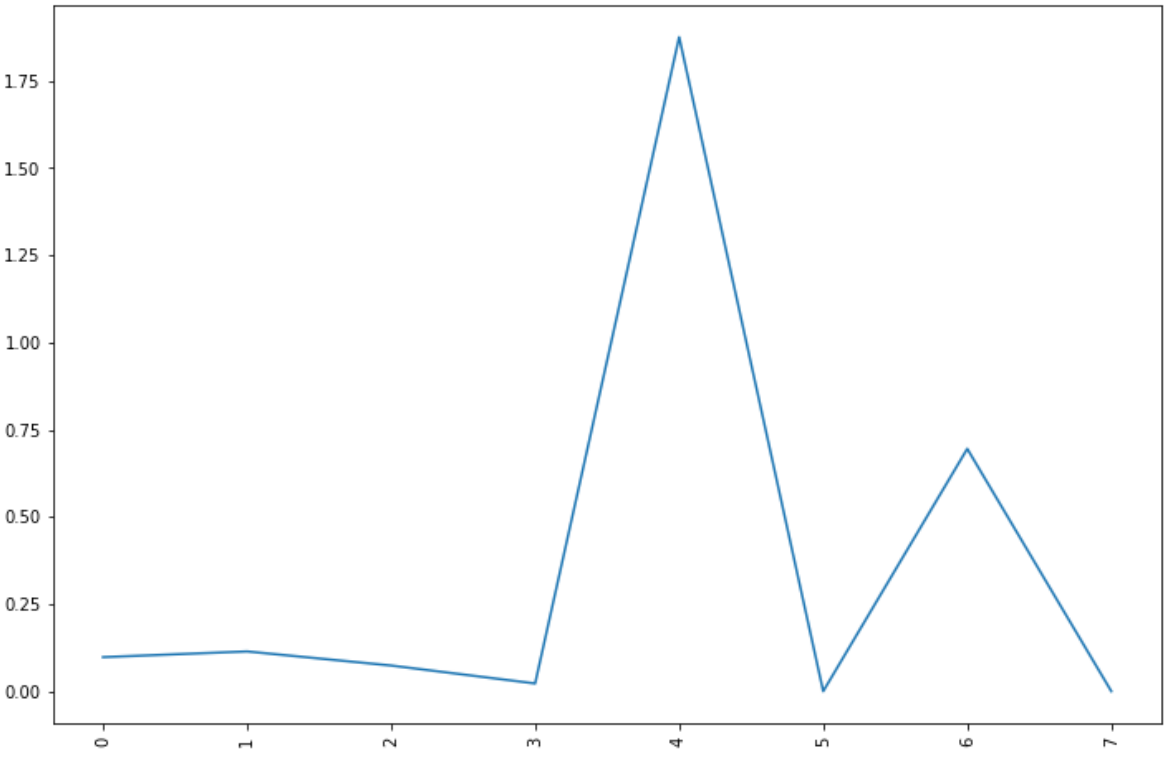
In [0]:

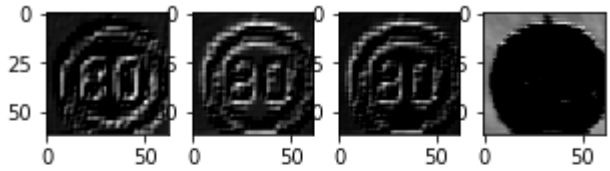
```
idx = 11722
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 2, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 4, 4, 1, y_true, y_prediction)
```

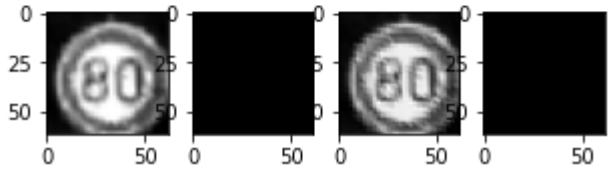
```
Actual class it belongs to: 5
Predicted class 2
Actual training image
```



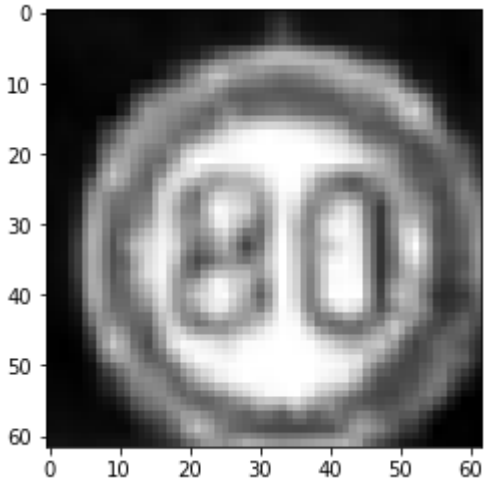

```
The kernel which activates/recognizes the shape: 4
```



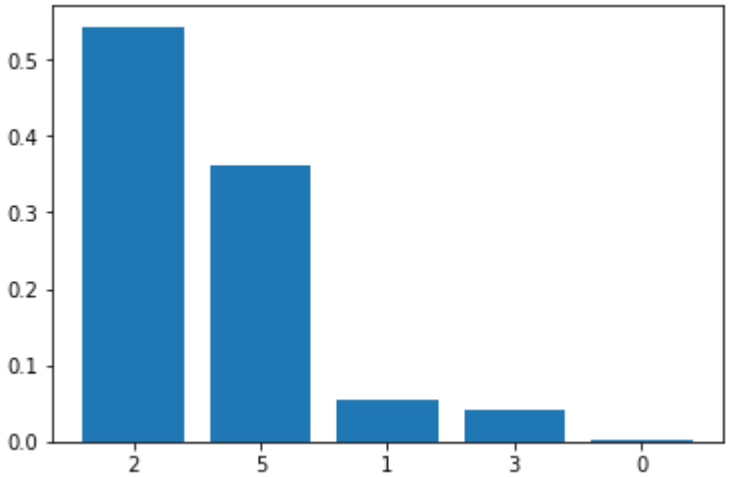




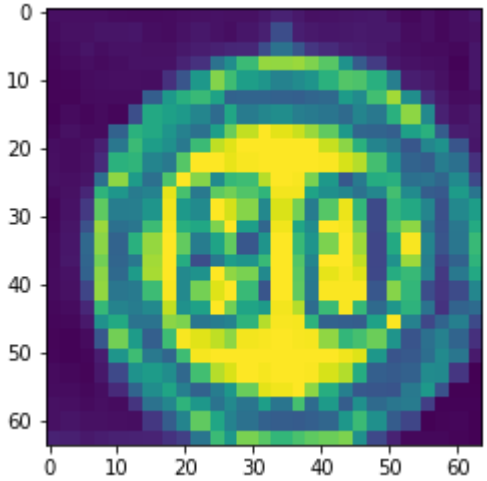
Multiple kernel activations starting from 0



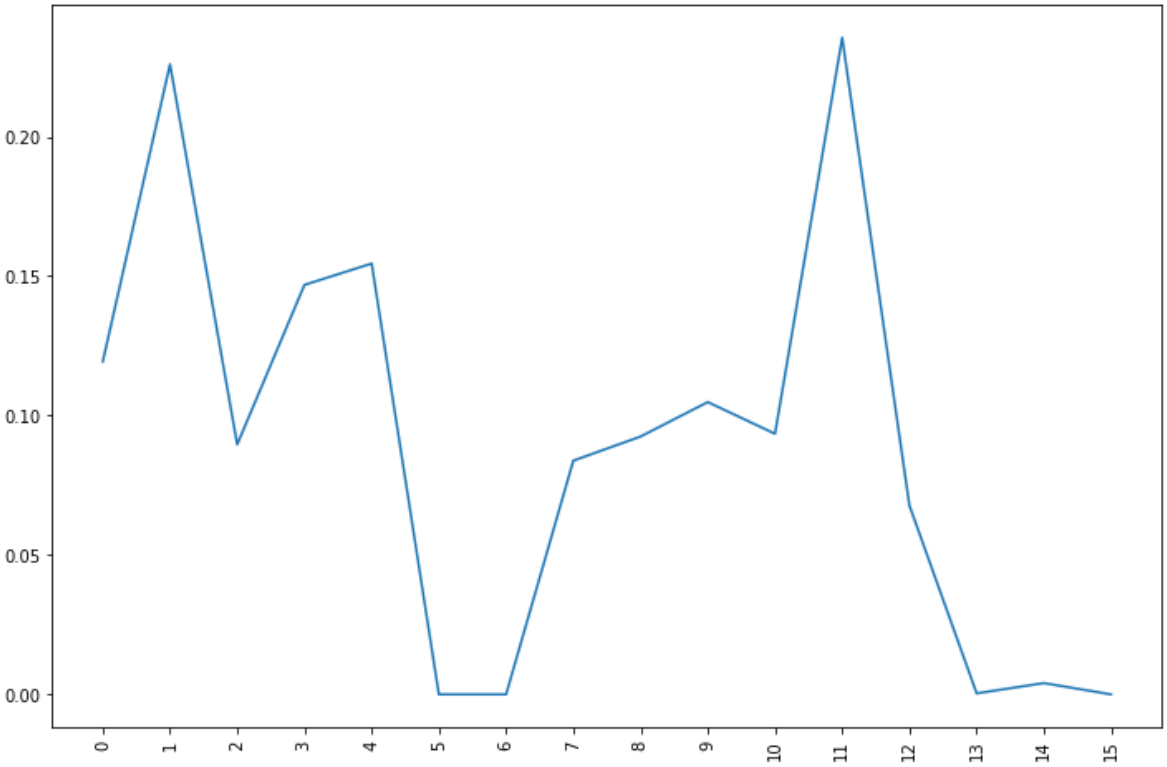
Activation for 4 kernel in 0 layer.

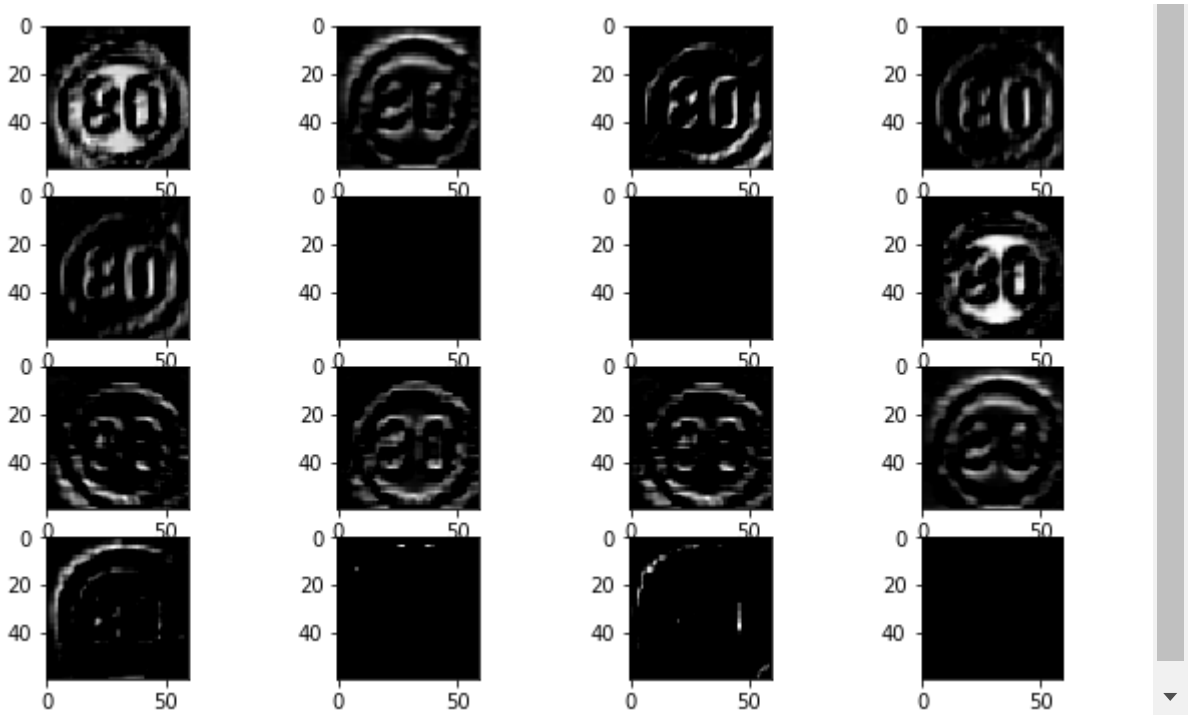


Probabilities of top 5 classes.
Actual class it belongs to: 5
Predicted class 2
Actual training image

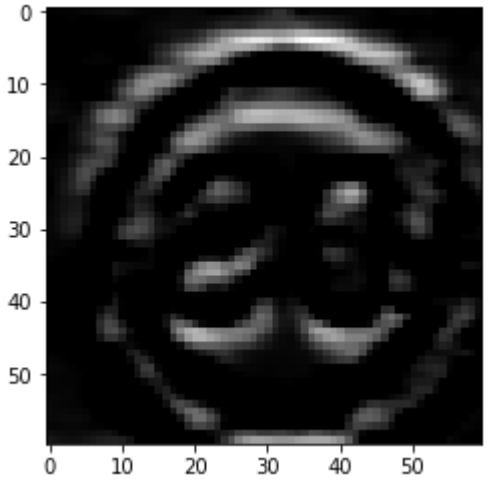



```
The kernel which activates/recognizes the shape: 11
```

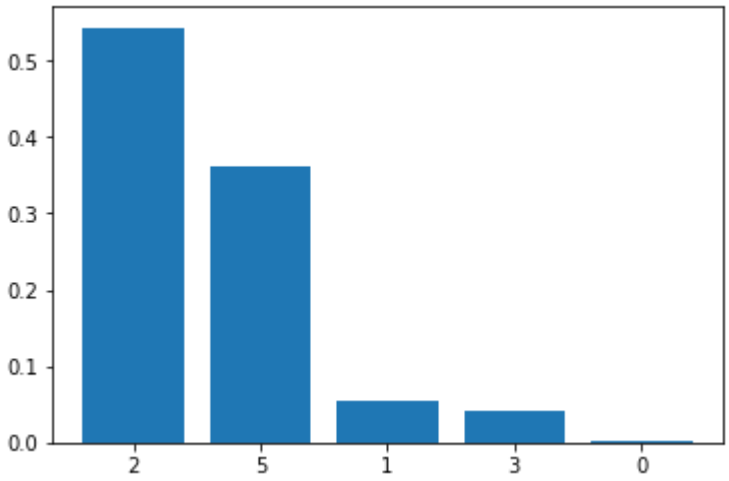




Multiple kernel activations starting from 0



Activation for 11 kernel in 1 layer.

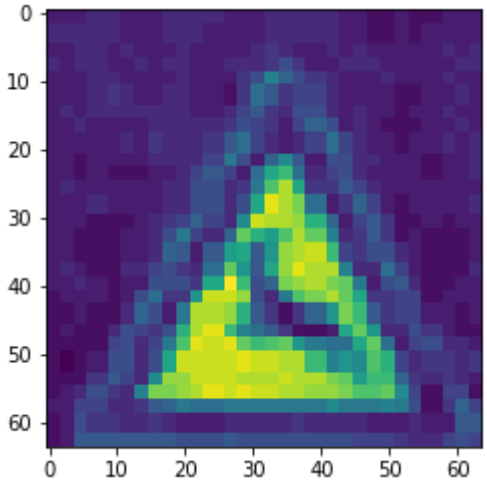


Probabilities of top 5 classes.

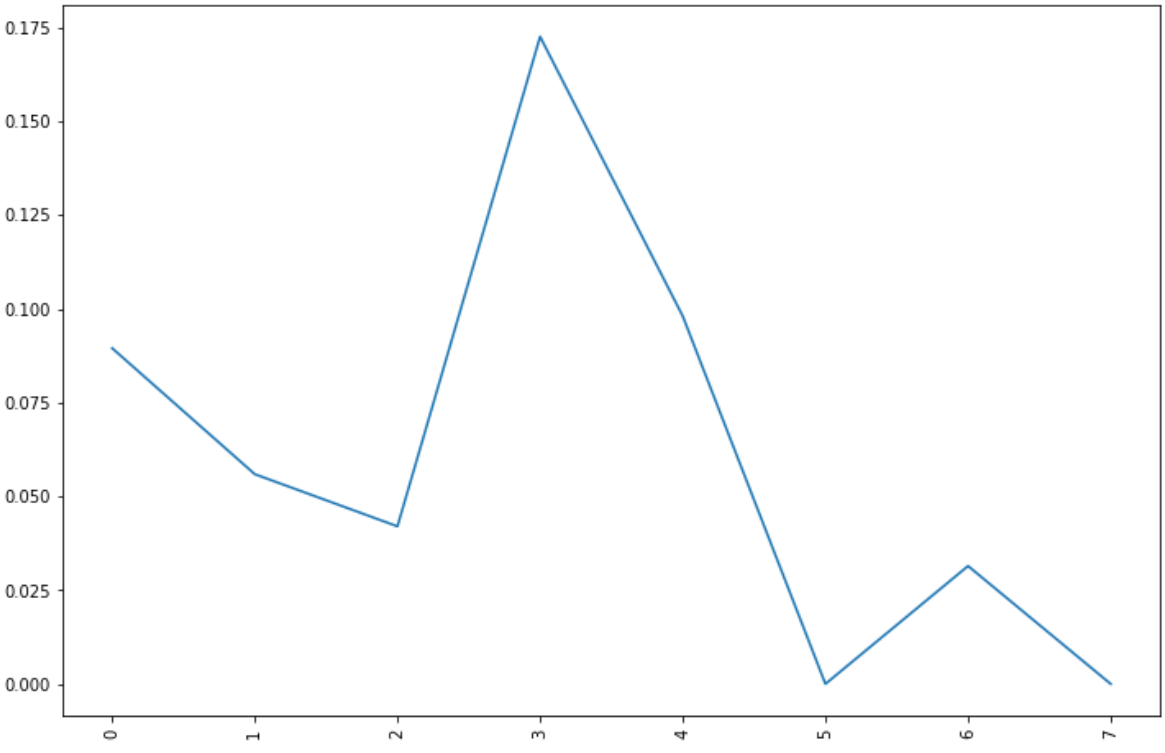
In [0]:

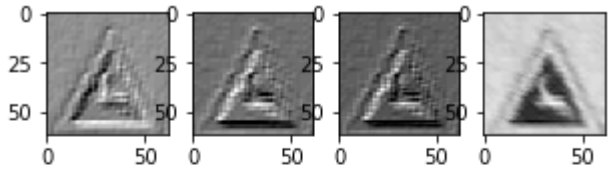
```
idx = 7861
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 2, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 4, 4, 1, y_true, y_prediction)
```

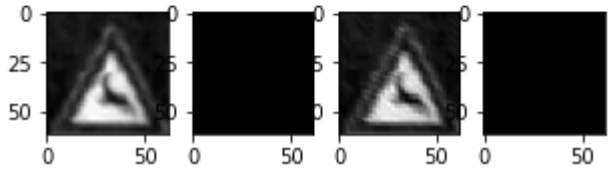
```
Actual class it belongs to: 31
Predicted class 26
Actual training image
```



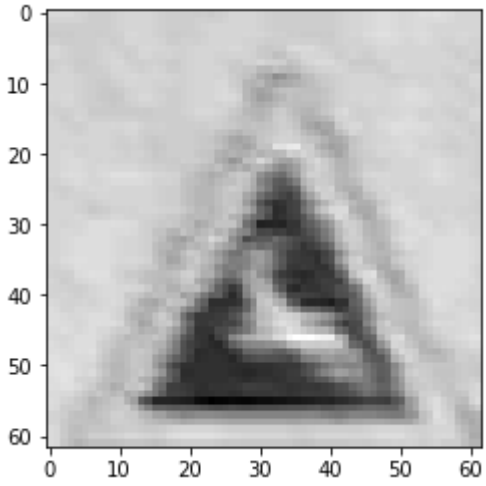

```
The kernel which activates/recognizes the shape: 3
```



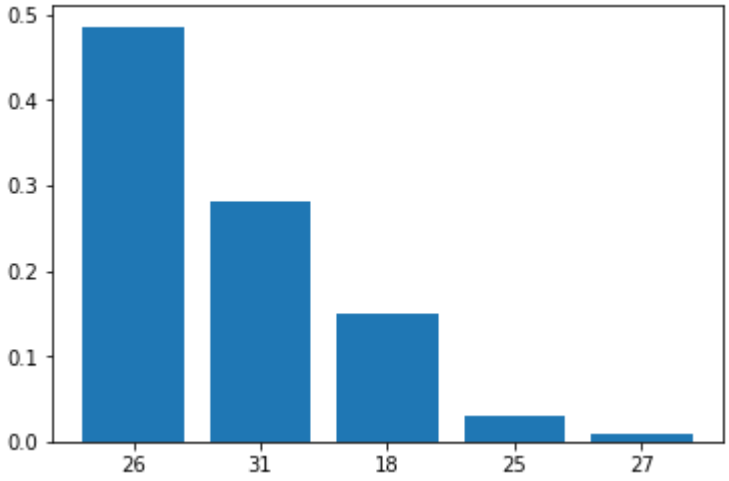




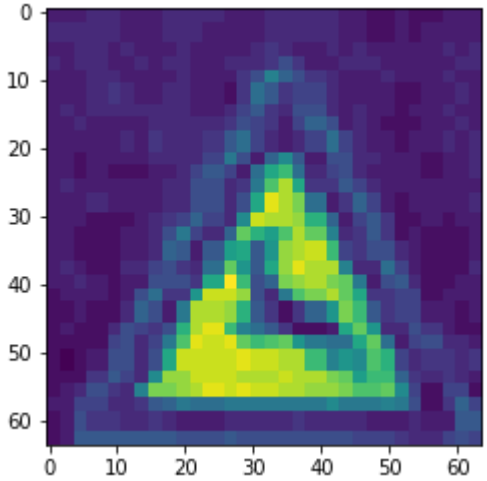
Multiple kernel activations starting from 0



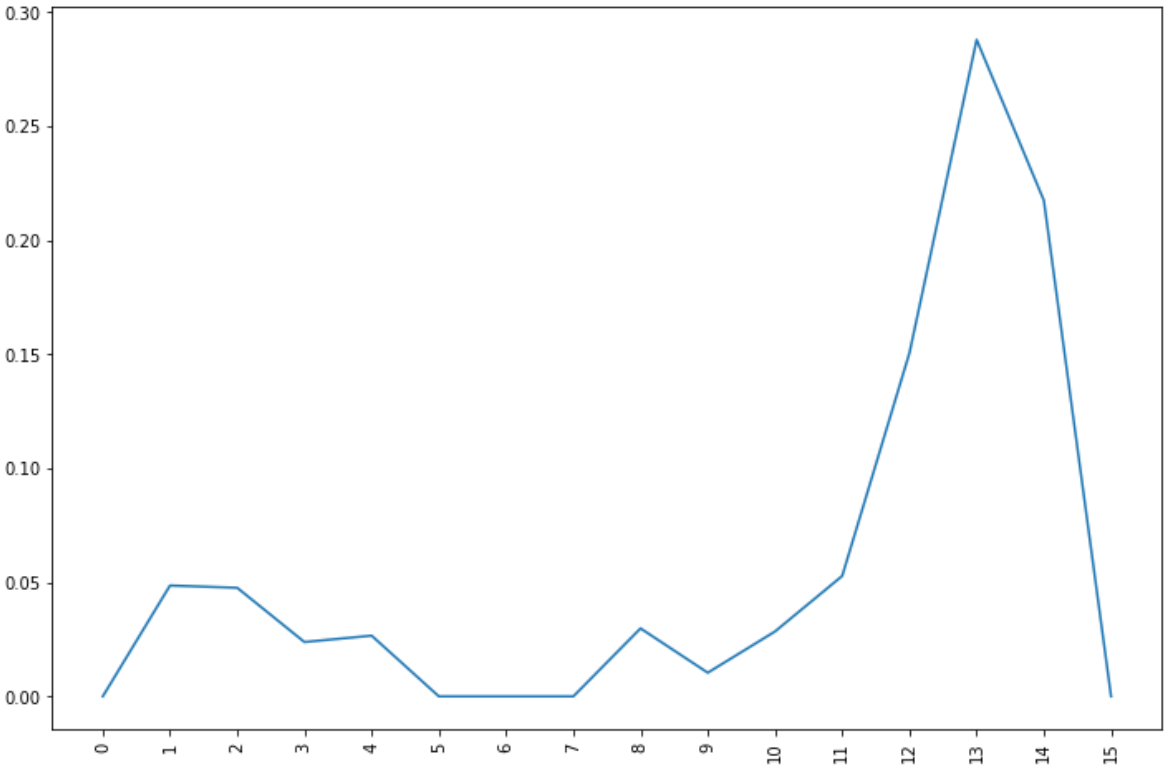
Activation for 3 kernel in 0 layer.

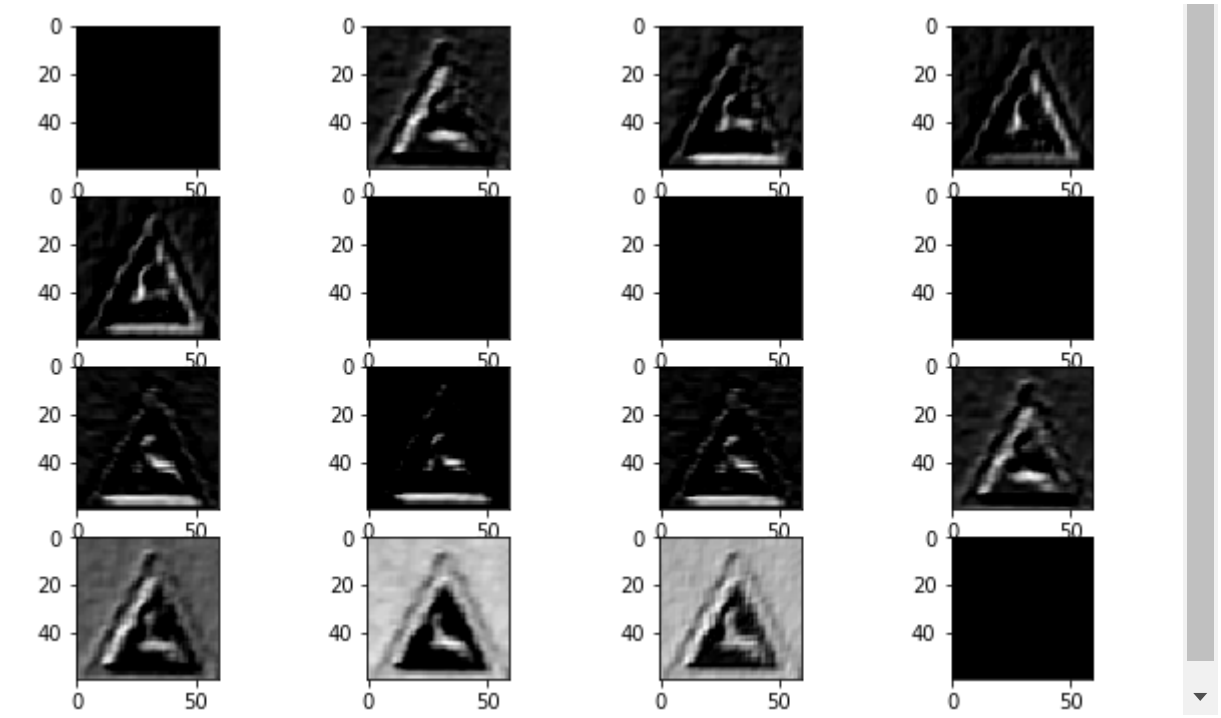


Probabilities of top 5 classes.
Actual class it belongs to: 31
Predicted class 26
Actual training image

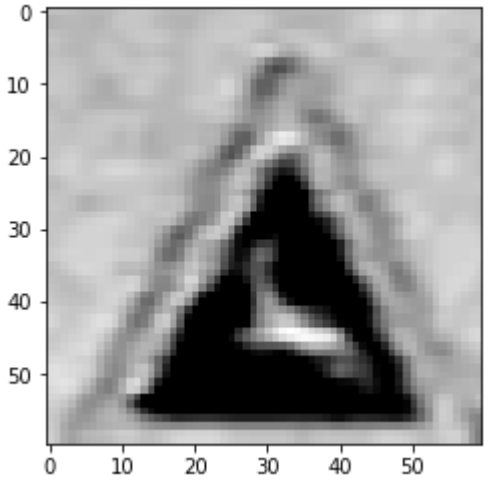



```
The kernel which activates/recognizes the shape: 13
```

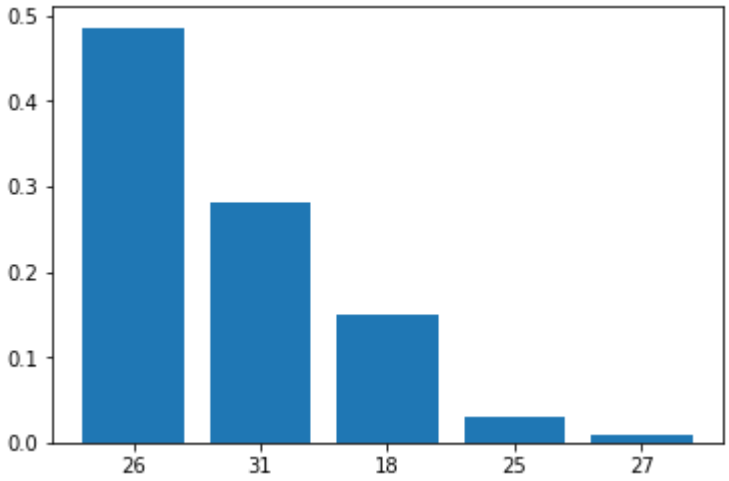




Multiple kernel activations starting from 0



Activation for 13 kernel in 1 layer.



Probabilities of top 5 classes.

[5.2] Conv(16 -- 3x3) - Conv(32 -- 3x3) - MaxPool(2x2) - Dropout(0.75) - Dense(128) - Dropout(0.5)

In [0]:

```
epochs = 15
model = Sequential()
model.add(Conv2D(
    16, kernel_size=(3, 3), activation='relu', input_shape=input_shape
))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.75))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(
    loss=keras.losses.categorical_crossentropy,
    optimizer=keras.optimizers.Adadelta(), metrics=['accuracy']
)

model.summary()

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 62, 62, 16)        160
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 60, 60, 32)        4640
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 28800)             0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               3686528
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 43)                5547
=================================================================
Total params: 3,696,875
Trainable params: 3,696,875
Non-trainable params: 0
_________________________________________________________________
Train on 27446 samples, validate on 11763 samples
Epoch 1/15
27446/27446 [==============================] - 6s 227us/step - loss: 1.9916
- acc: 0.4784 - val_loss: 0.5706 - val_acc: 0.8239
Epoch 2/15
27446/27446 [==============================] - 5s 183us/step - loss: 0.6065
- acc: 0.8301 - val_loss: 0.2726 - val_acc: 0.9351
Epoch 3/15
27446/27446 [==============================] - 5s 184us/step - loss: 0.4118
- acc: 0.8843 - val_loss: 0.1838 - val_acc: 0.9569
Epoch 4/15
```

```
27446/27446 [==============================] - 5s 183us/step - loss: 0.3258
- acc: 0.9074 - val_loss: 0.1454 - val_acc: 0.9664
Epoch 5/15
27446/27446 [==============================] - 5s 184us/step - loss: 0.2832
- acc: 0.9200 - val_loss: 0.1278 - val_acc: 0.9681
Epoch 6/15
27446/27446 [==============================] - 5s 184us/step - loss: 0.2383
- acc: 0.9321 - val_loss: 0.1113 - val_acc: 0.9740
Epoch 7/15
27446/27446 [==============================] - 5s 184us/step - loss: 0.2182
- acc: 0.9369 - val_loss: 0.1048 - val_acc: 0.9757
Epoch 8/15
27446/27446 [==============================] - 5s 185us/step - loss: 0.1981
- acc: 0.9420 - val_loss: 0.0957 - val_acc: 0.9763
Epoch 9/15
27446/27446 [==============================] - 5s 185us/step - loss: 0.1825
- acc: 0.9461 - val_loss: 0.0892 - val_acc: 0.9783
Epoch 10/15
27446/27446 [==============================] - 5s 188us/step - loss: 0.1672
- acc: 0.9506 - val_loss: 0.0822 - val_acc: 0.9793
Epoch 11/15
27446/27446 [==============================] - 5s 189us/step - loss: 0.1535
- acc: 0.9545 - val_loss: 0.0825 - val_acc: 0.9797
Epoch 12/15
27446/27446 [==============================] - 5s 186us/step - loss: 0.1467
- acc: 0.9557 - val_loss: 0.0747 - val_acc: 0.9833
Epoch 13/15
27446/27446 [==============================] - 5s 186us/step - loss: 0.1351
- acc: 0.9598 - val_loss: 0.0729 - val_acc: 0.9817
Epoch 14/15
27446/27446 [==============================] - 5s 185us/step - loss: 0.1312
- acc: 0.9588 - val_loss: 0.0694 - val_acc: 0.9827
Epoch 15/15
27446/27446 [==============================] - 5s 185us/step - loss: 0.1265
- acc: 0.9625 - val_loss: 0.0742 - val_acc: 0.9829
```

In [0]:

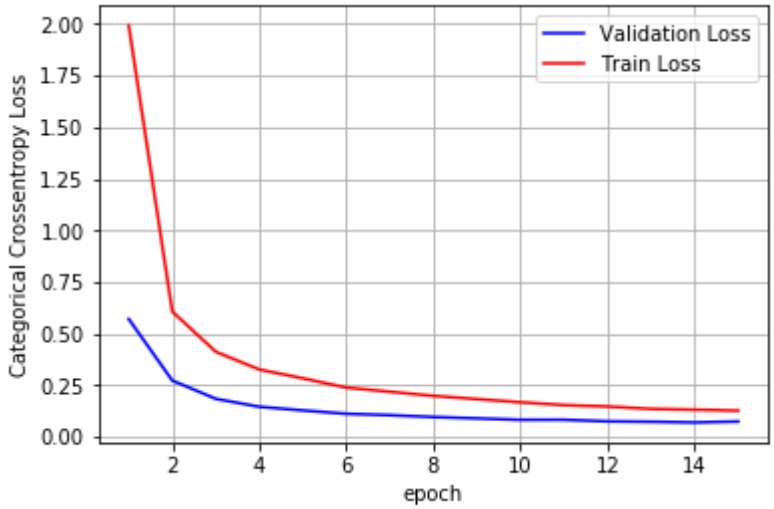
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(fig, x, vy, ty, ax)
```

```
Test score: 0.07415205248816249
Test accuracy: 0.9829125223157358
```



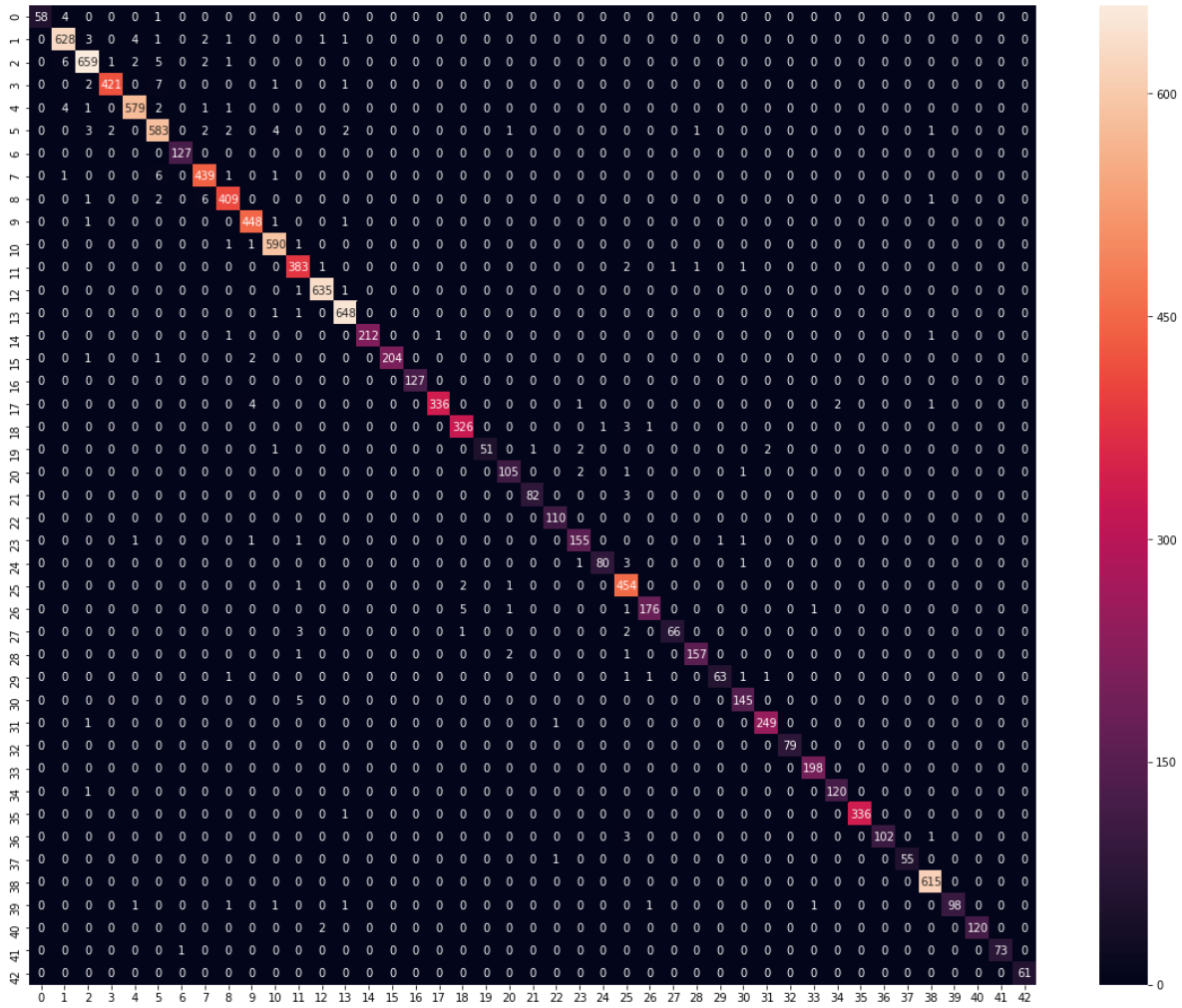
In [0]:

```
plt.figure(figsize=(20,16))
y_prediction = model.predict(x_test)
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_prediction, axis = 1)
# Convert validation observations to one hot vectors
y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
print("-----------------------------------------------------------")
print("Micro F1 score", f1_score(y_true, y_pred_classes, average='micro'))
print("-----------------------------------------------------------")
confusion_mtx = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(confusion_mtx, annot=True, fmt="d")
```

```
-----------------------------------------------------------
Micro F1 score 0.9829125223157358
-----------------------------------------------------------
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b9f579d30>
```



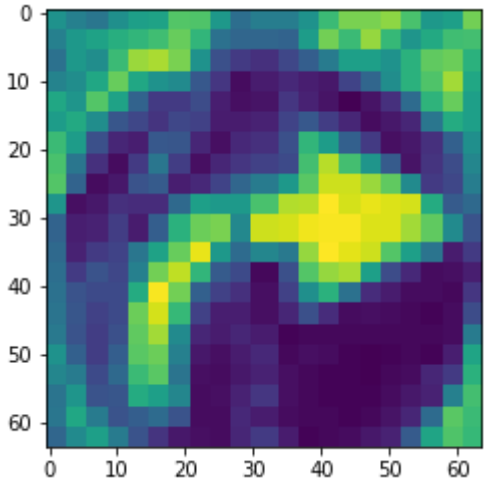
In [0]:

```
layer_outputs = [layer.output for layer in model.layers]
activation_model = Model(inputs=model.input, outputs=layer_outputs)
```

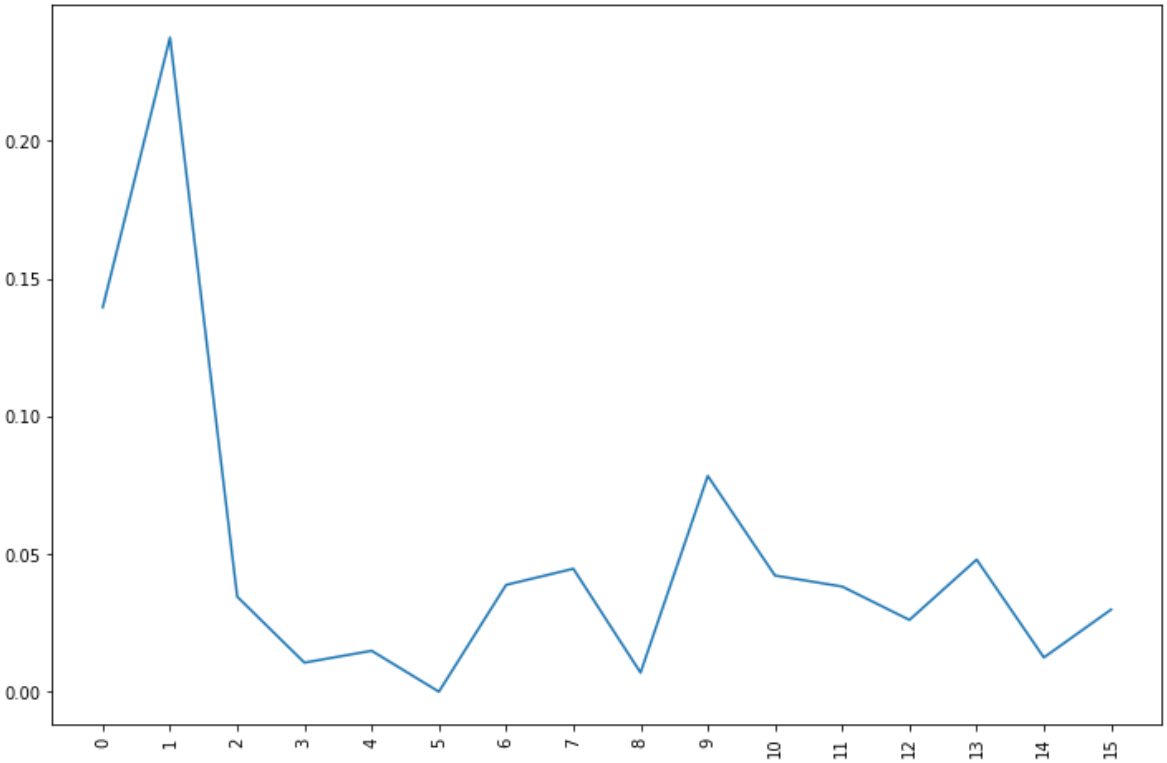
In [0]:

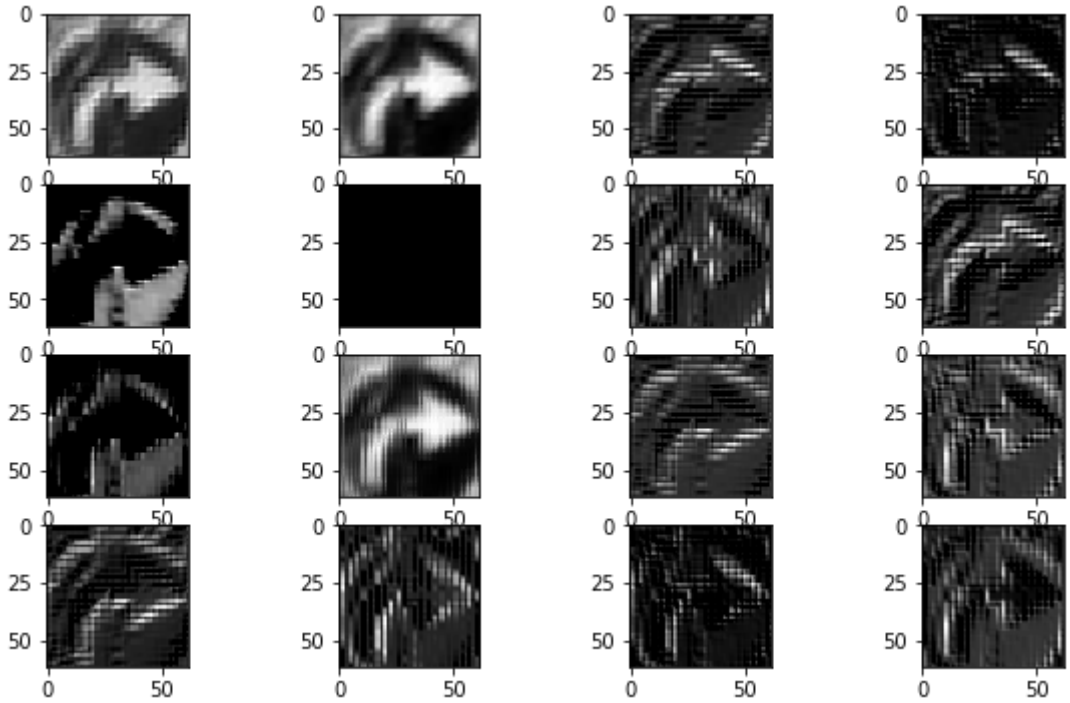
```
idx = 9942
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 33
Predicted class 33
Actual training image
```

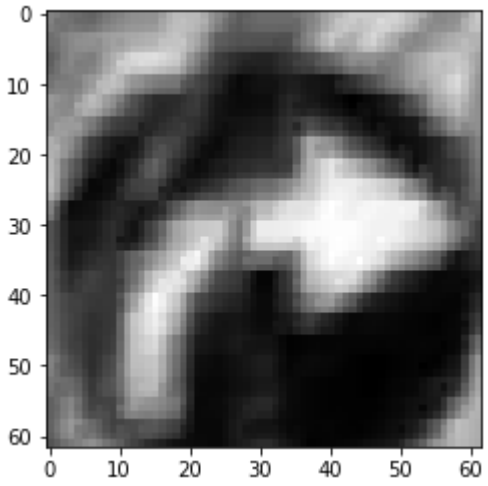



```
The kernel which activates/recognizes the shape: 1
```

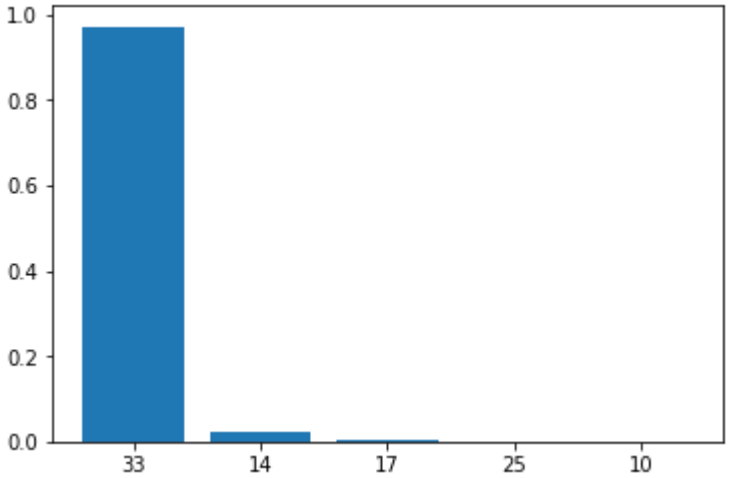




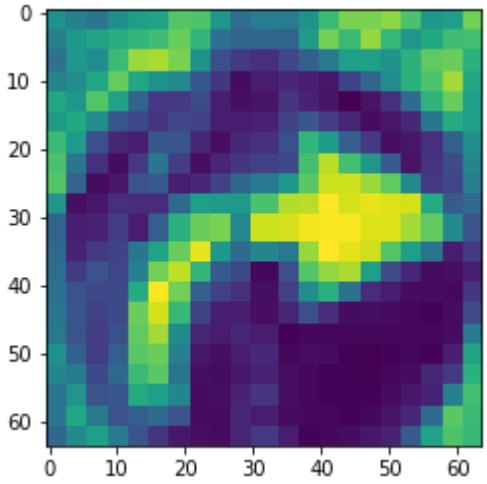
Multiple kernel activations starting from 0



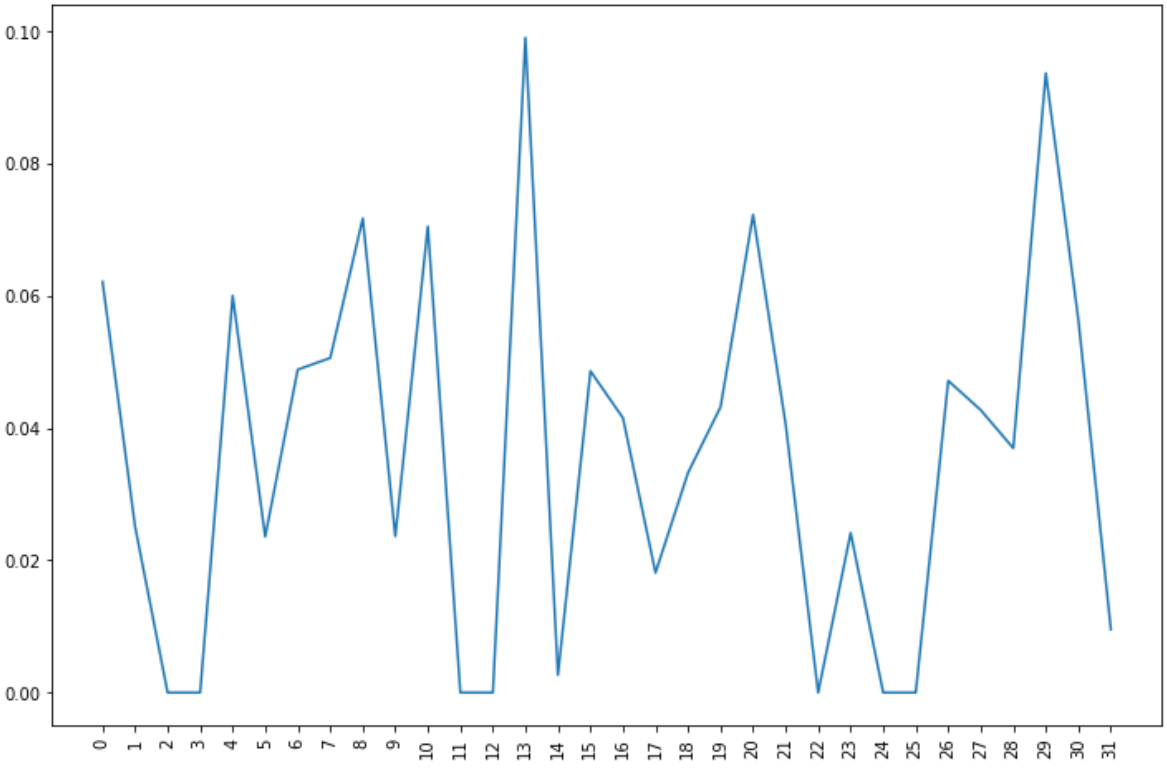
Activation for 1 kernel in 0 layer.

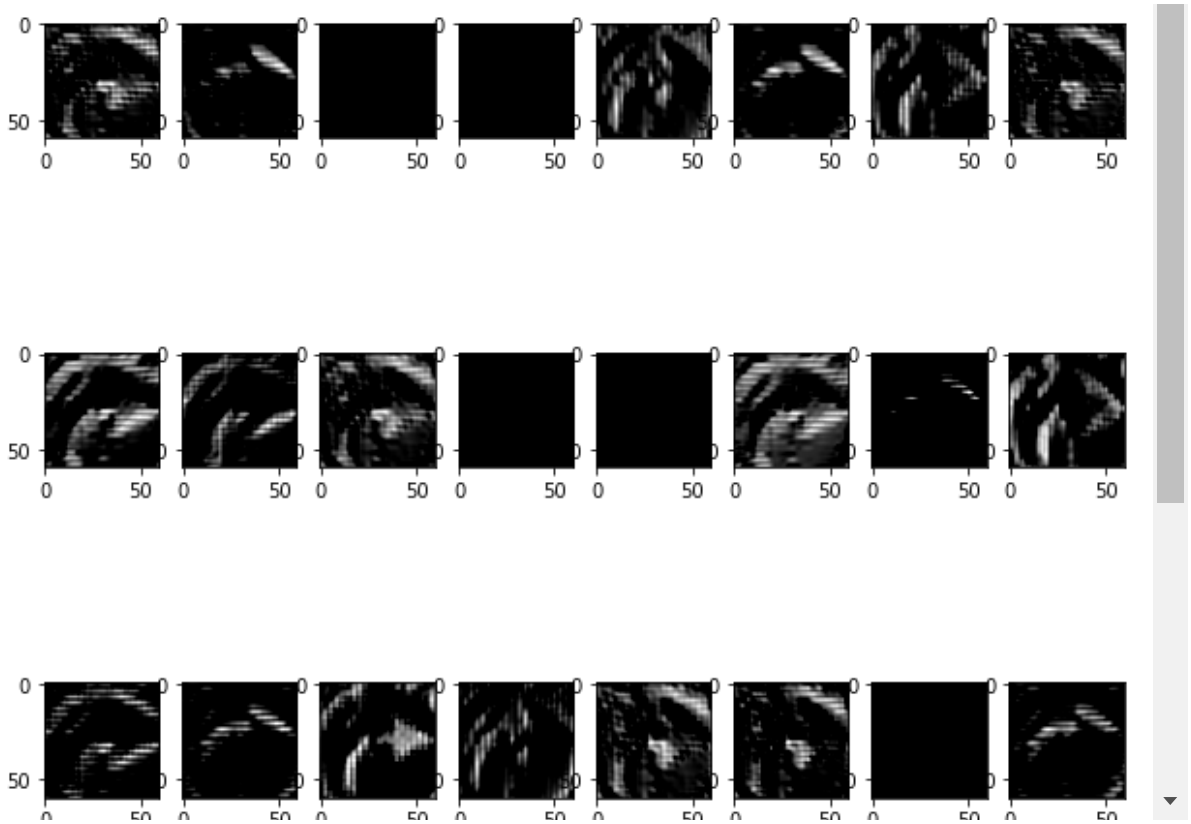


Probabilities of top 5 classes.
Actual class it belongs to: 33
Predicted class 33
Actual training image

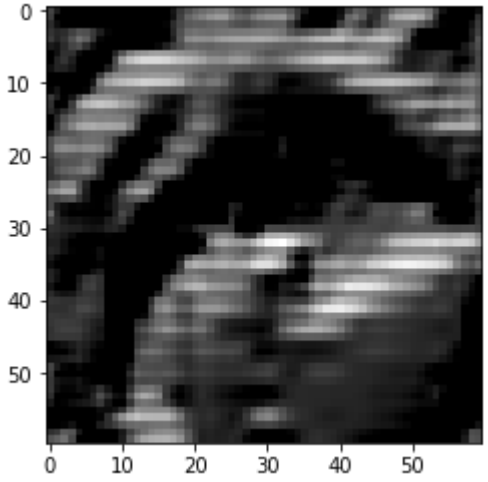



```
The kernel which activates/recognizes the shape: 13
```

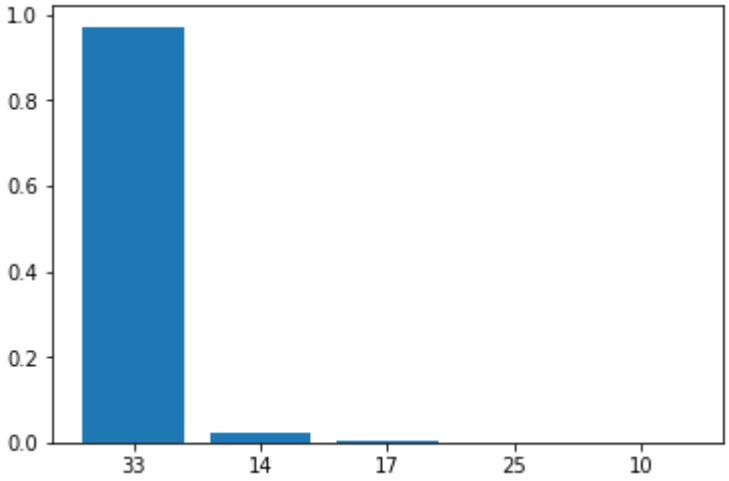




Multiple kernel activations starting from 0



Activation for 13 kernel in 1 layer.

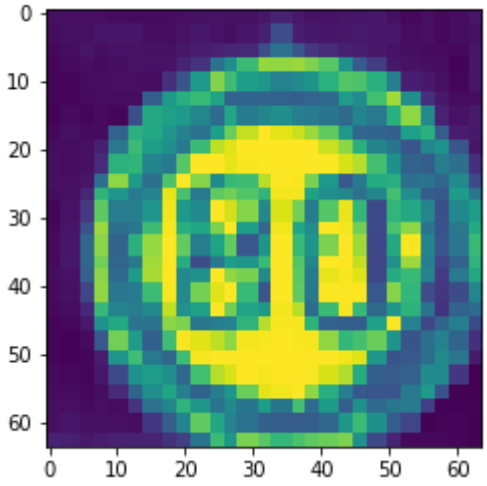


Probabilities of top 5 classes.

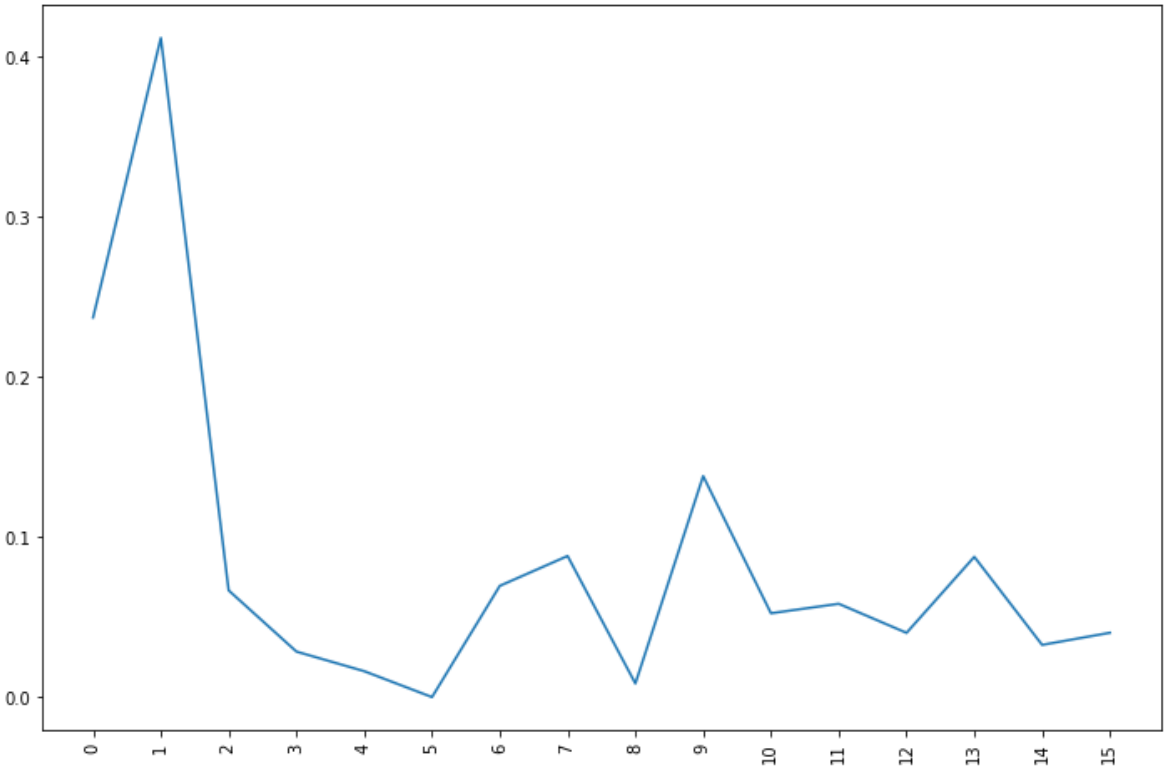
In [0]:

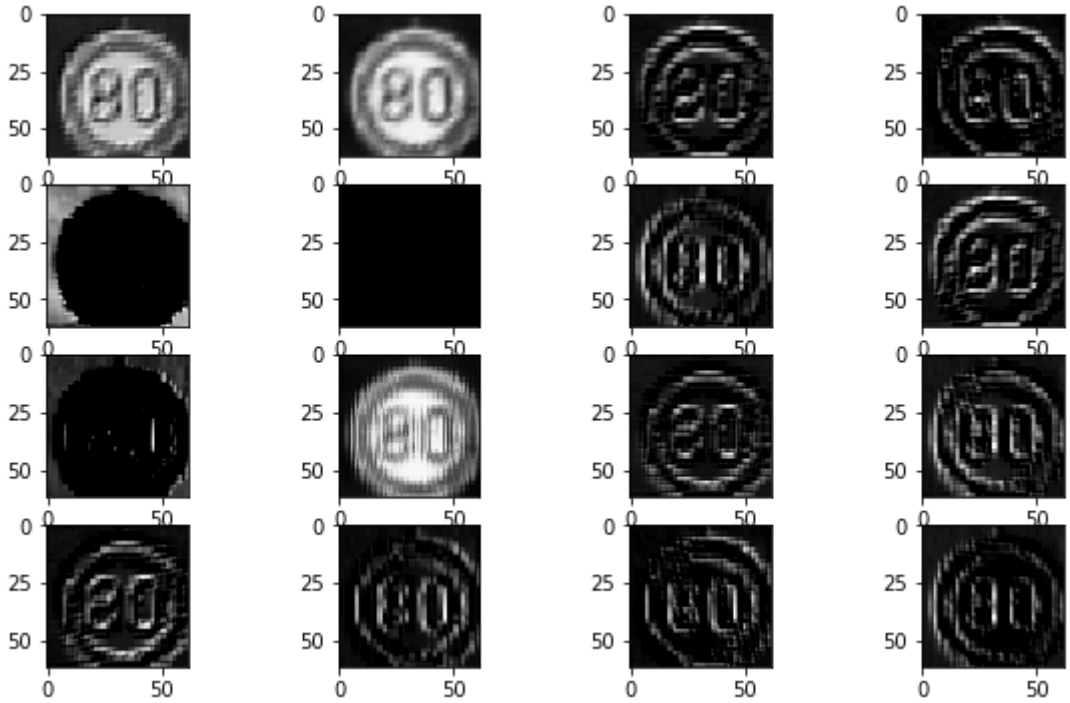
```
idx = 11722
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 5
Predicted class 5
Actual training image
```

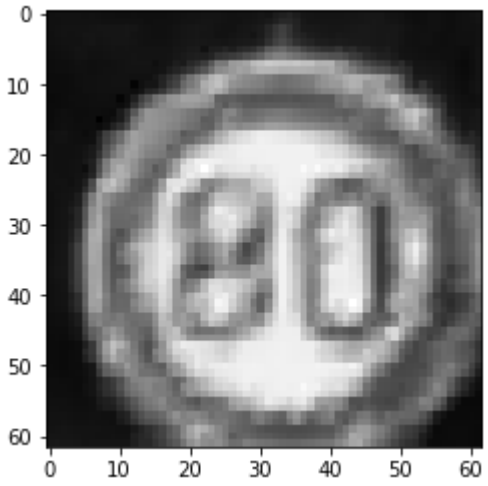



```
The kernel which activates/recognizes the shape: 1
```

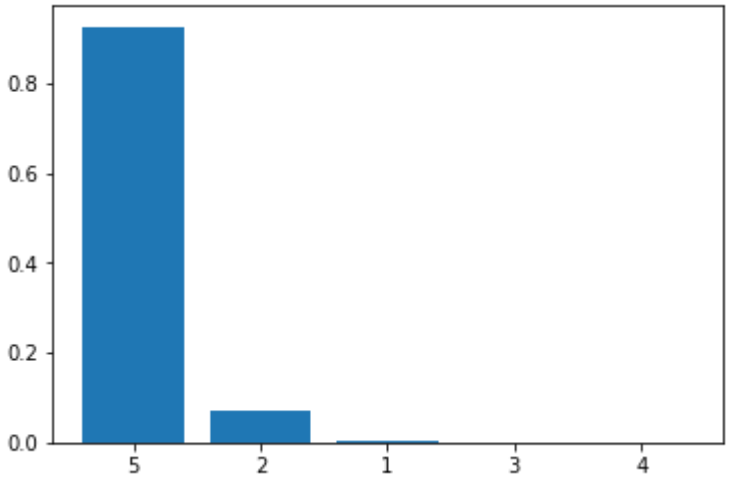




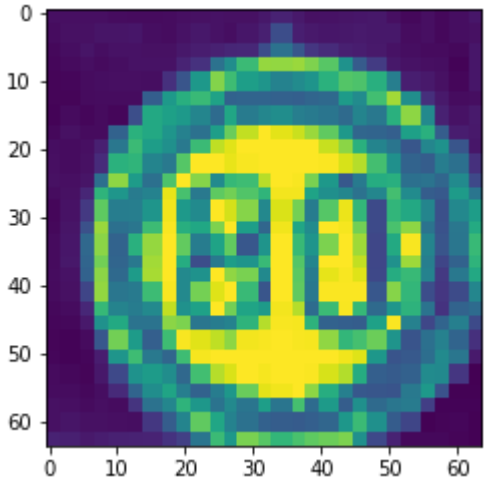
Multiple kernel activations starting from 0



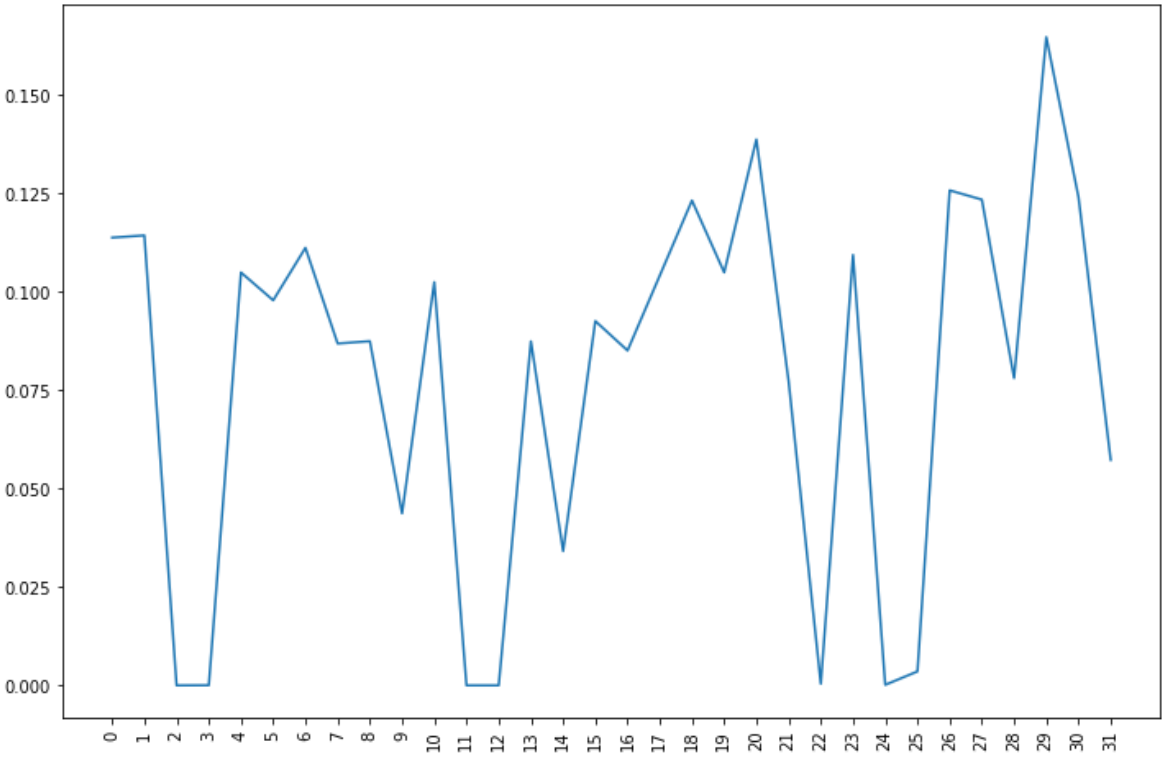
Activation for 1 kernel in 0 layer.

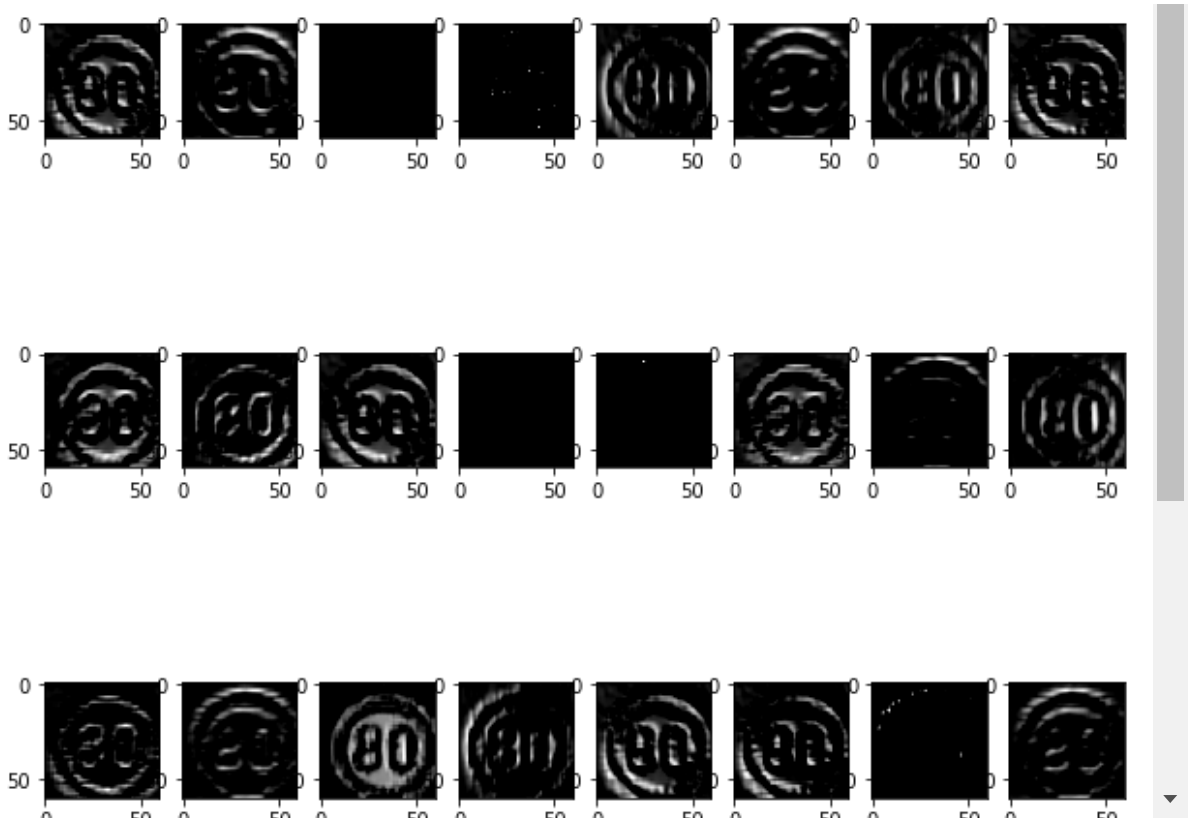


Probabilities of top 5 classes.
Actual class it belongs to: 5
Predicted class 5
Actual training image

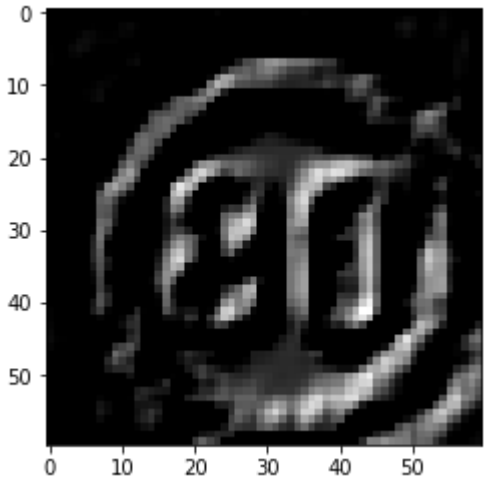



```
The kernel which activates/recognizes the shape: 29
```

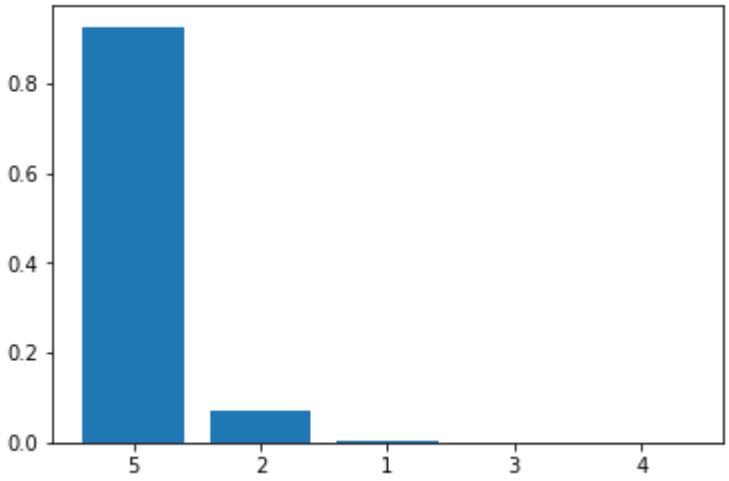




Multiple kernel activations starting from 0



Activation for 29 kernel in 1 layer.

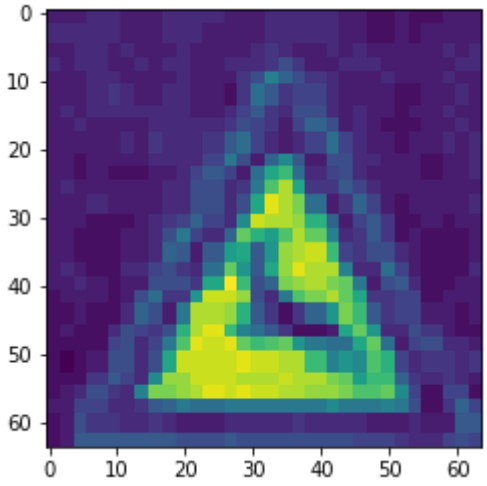


Probabilities of top 5 classes.

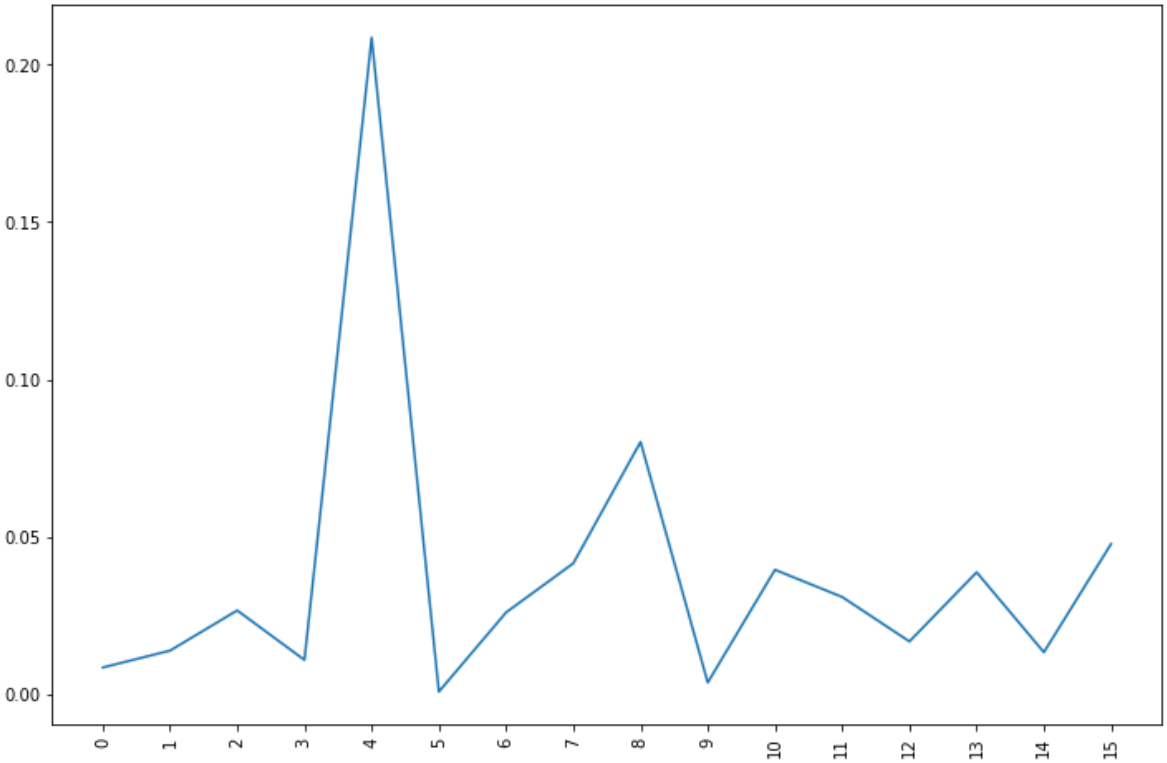
In [0]:

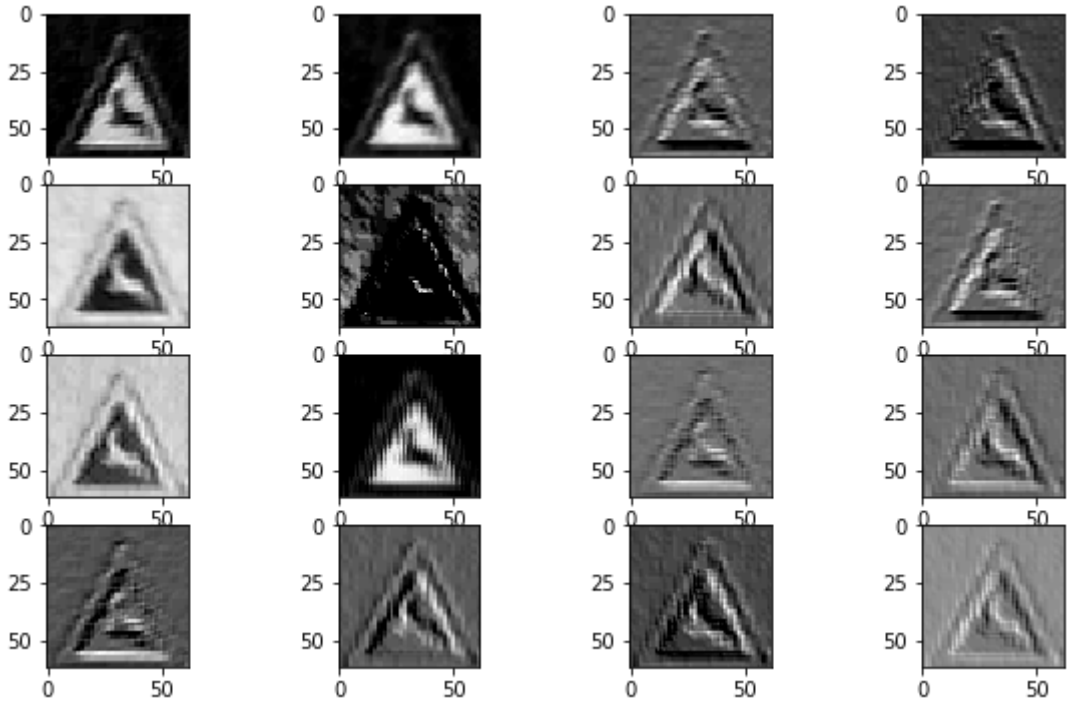
```
idx = 7861
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 31
Predicted class 31
Actual training image
```

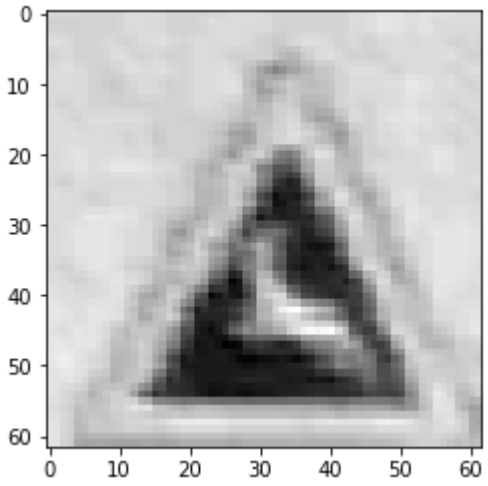



```
The kernel which activates/recognizes the shape: 4
```

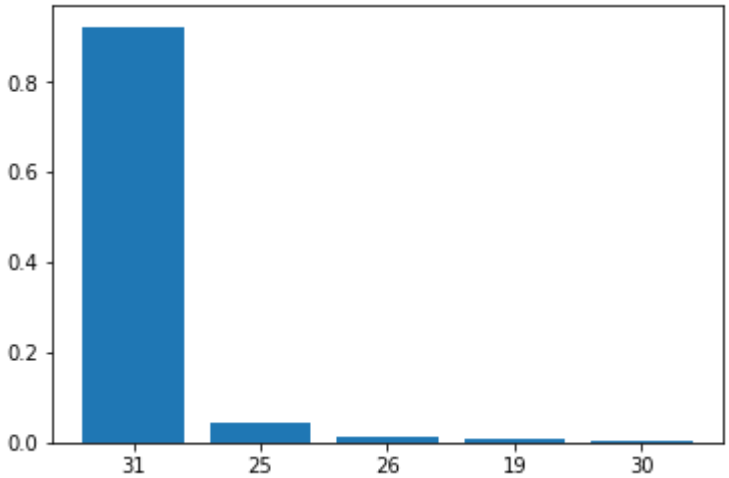




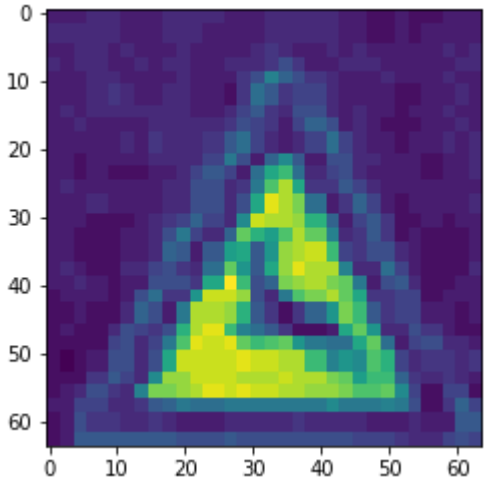
Multiple kernel activations starting from 0



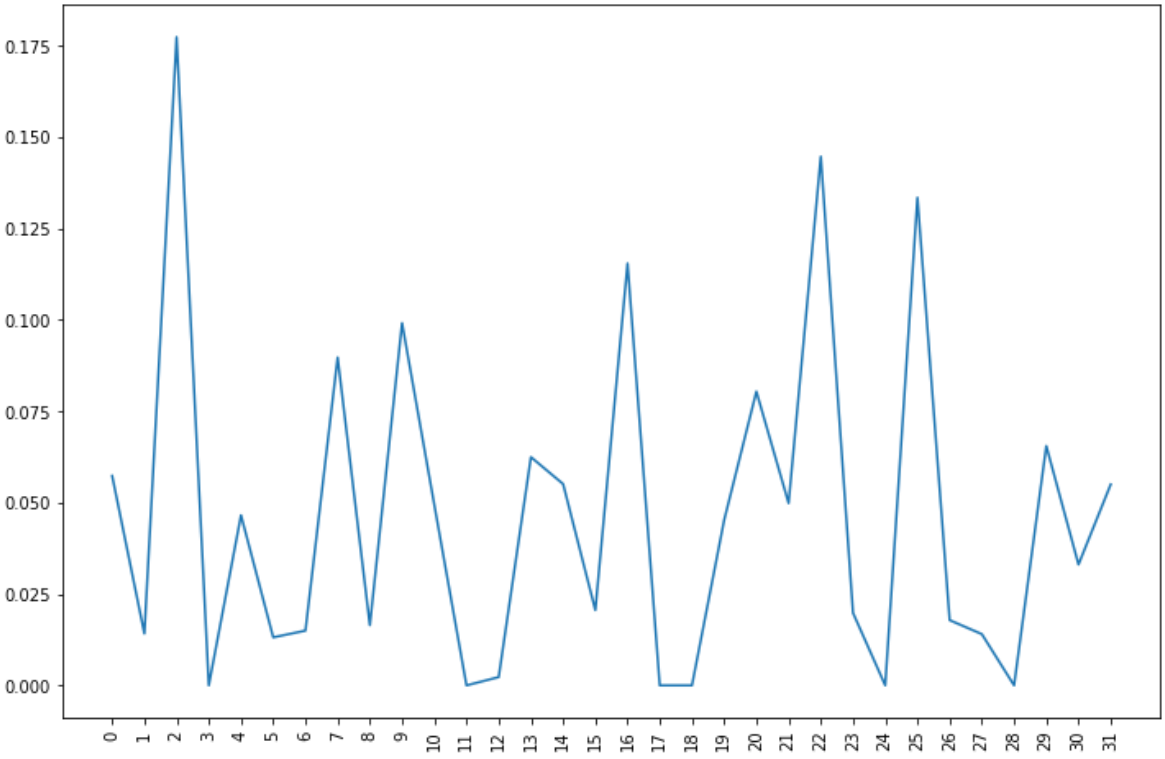
Activation for 4 kernel in 0 layer.

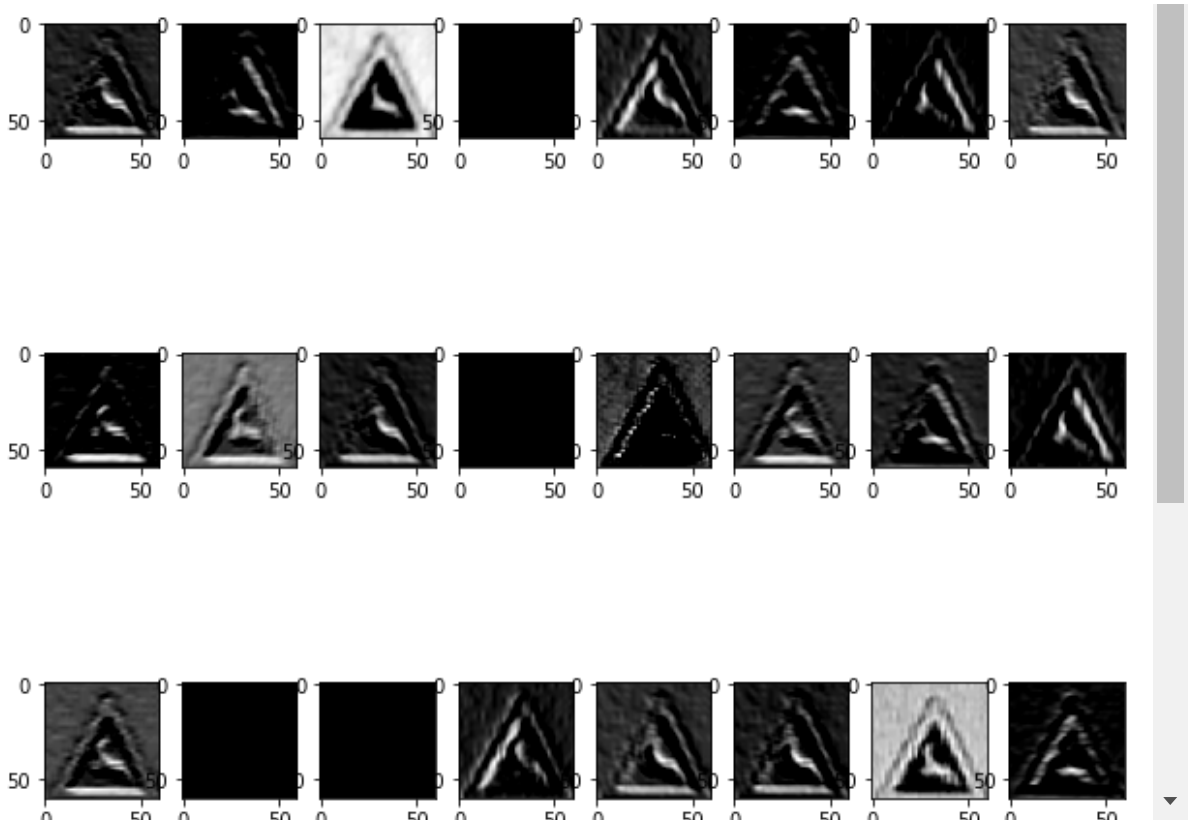


Probabilities of top 5 classes.
Actual class it belongs to: 31
Predicted class 31
Actual training image

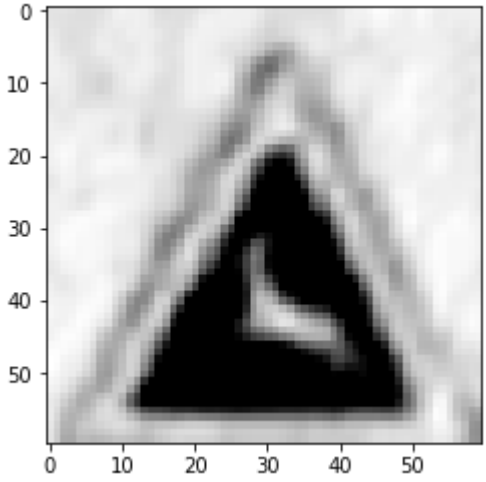



```
The kernel which activates/recognizes the shape: 2
```

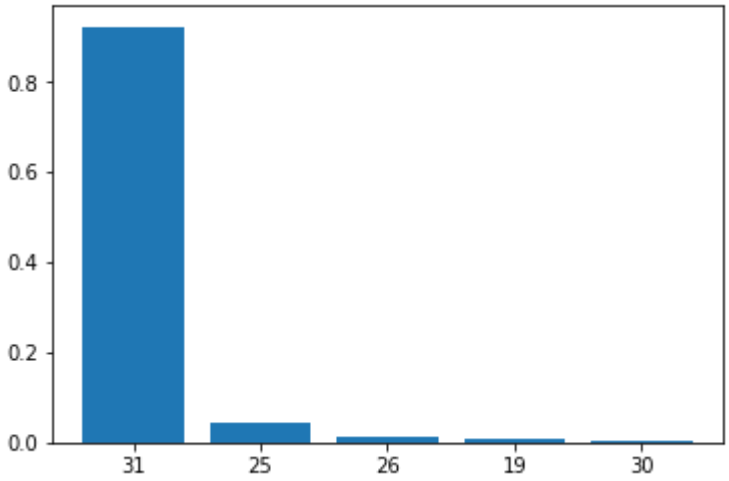




Multiple kernel activations starting from 0



Activation for 2 kernel in 1 layer.



Probabilities of top 5 classes.

In [0]:

```
actual_class = []
index_image = []
predicted_class = []
for idx, im in enumerate(y_true):
    if im != np.argmax(y_prediction[idx]):
        actual_class.append(im)
        predicted_class.append(np.argmax(y_prediction[idx]))
        index_image.append(idx)
np.array(index_image)
```

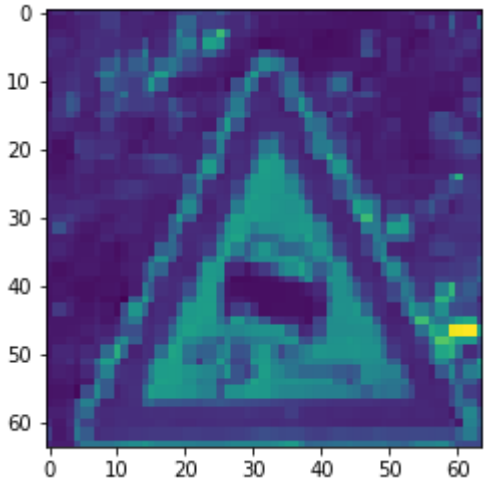
Out[62]:

```
array([   20,    41,   126,   157,   496,   522,   569,   579,   592,
         621,   696,   711,   836,   882,   981,  1009,  1053,  1148,
        1152,  1188,  1193,  1210,  1275,  1321,  1391,  1478,  1485,
        1578,  1609,  1689,  1750,  1774,  1840,  1880,  1915,  1926,
        1993,  1998,  2082,  2288,  2361,  2552,  2657,  2722,  2791,
        2873,  2925,  2978,  3050,  3055,  3080,  3113,  3226,  3290,
        3319,  3441,  3450,  3470,  3509,  3510,  3513,  3551,  3576,
        3592,  3629,  3665,  3669,  3846,  3853,  3962,  4028,  4083,
        4101,  4157,  4302,  4316,  4340,  4375,  4558,  4629,  4646,
        4781,  4921,  4926,  4953,  4975,  5032,  5082,  5089,  5223,
        5240,  5269,  5446,  5491,  5513,  5666,  5677,  5802,  5818,
        5855,  6018,  6028,  6037,  6102,  6112,  6128,  6157,  6166,
        6167,  6175,  6324,  6384,  6464,  6483,  6669,  6684,  6795,
        6806,  6857,  6930,  7072,  7086,  7269,  7271,  7358,  7378,
        7436,  7481,  7496,  7520,  7521,  7570,  7626,  7700,  7796,
        7871,  7941,  7954,  8047,  8090,  8174,  8181,  8200,  8207,
        8227,  8235,  8346,  8348,  8412,  8492,  8510,  8525,  8542,
        8578,  8627,  8632,  8668,  8677,  8719,  8723,  8739,  8876,
        8887,  8945,  9058,  9317,  9376,  9477,  9584,  9749,  9873,
       10098, 10125, 10554, 10587, 10589, 10605, 10884, 10917, 10943,
       10969, 10972, 11018, 11042, 11063, 11090, 11120, 11167, 11211,
       11301, 11307, 11333, 11354, 11383, 11404, 11432, 11483, 11527,
       11653, 11684, 11711])
```

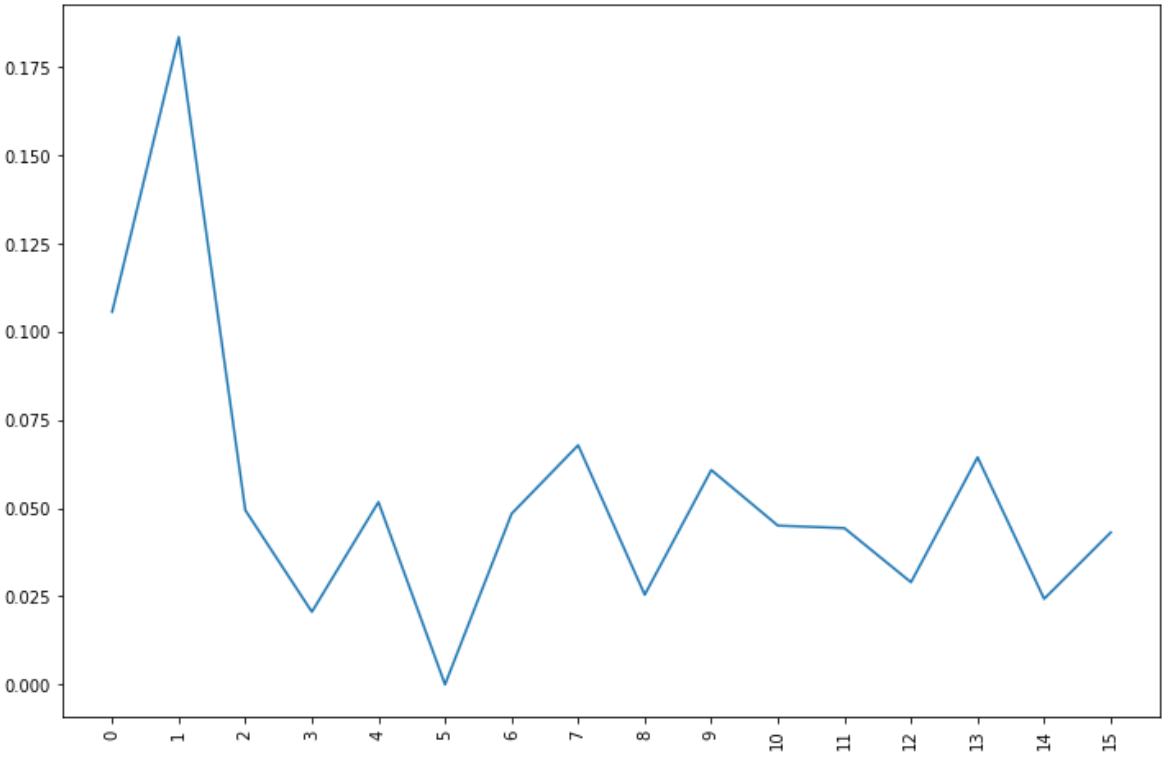
In [0]:

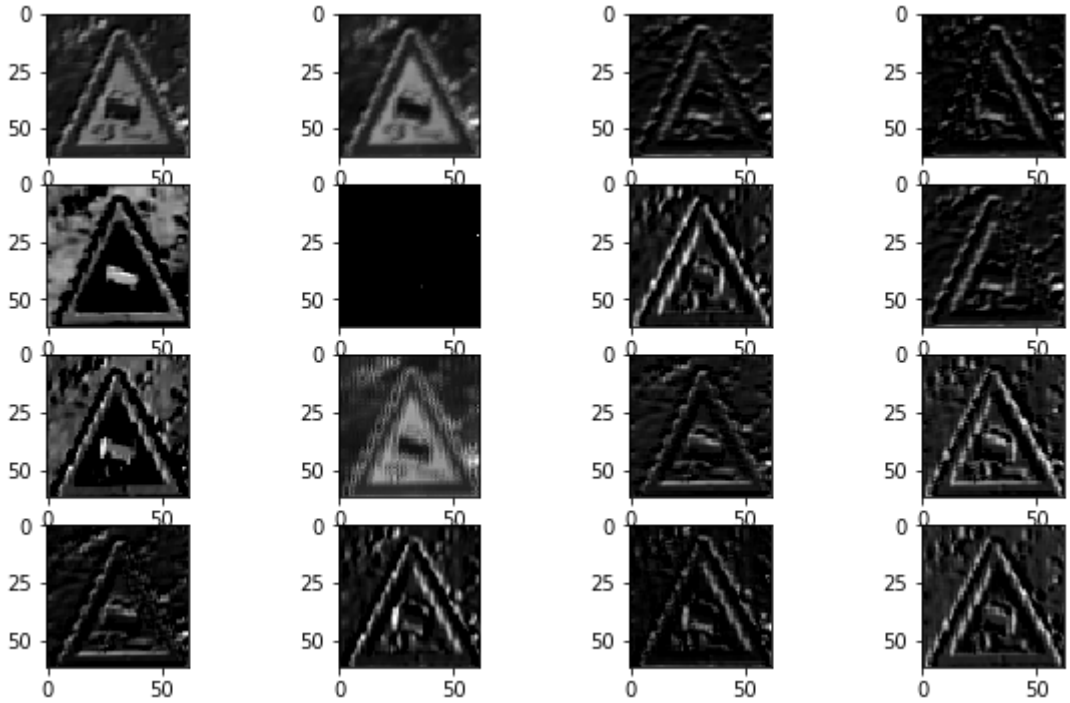
```
idx = 711
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 23
Predicted class 30
Actual training image
```

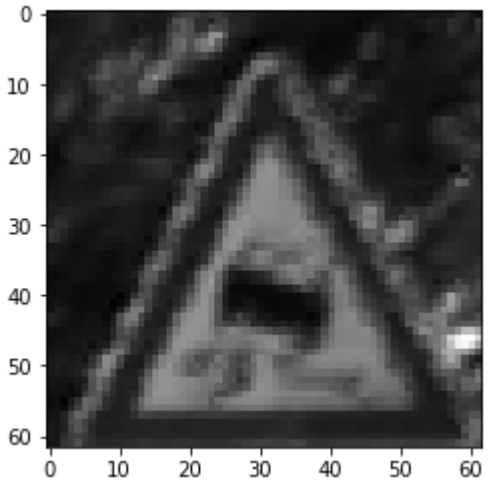



```
The kernel which activates/recognizes the shape: 1
```

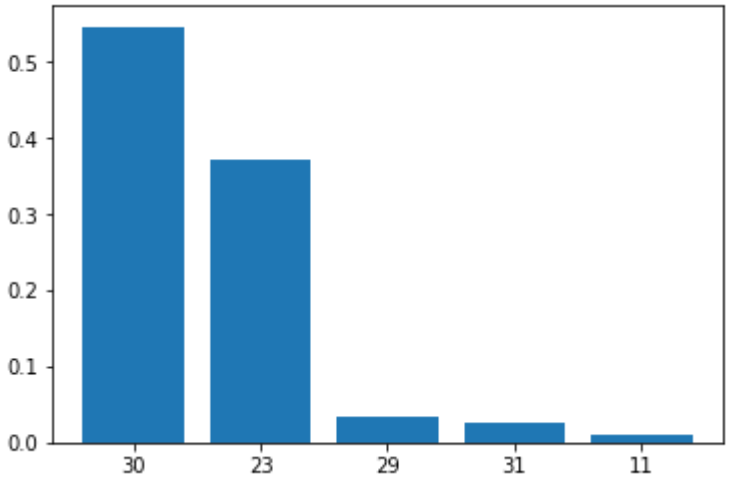




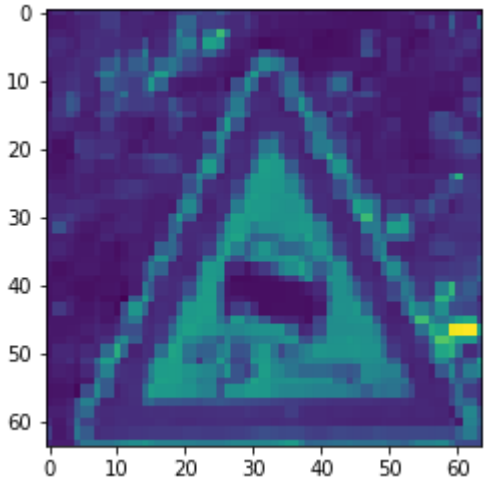
Multiple kernel activations starting from 0



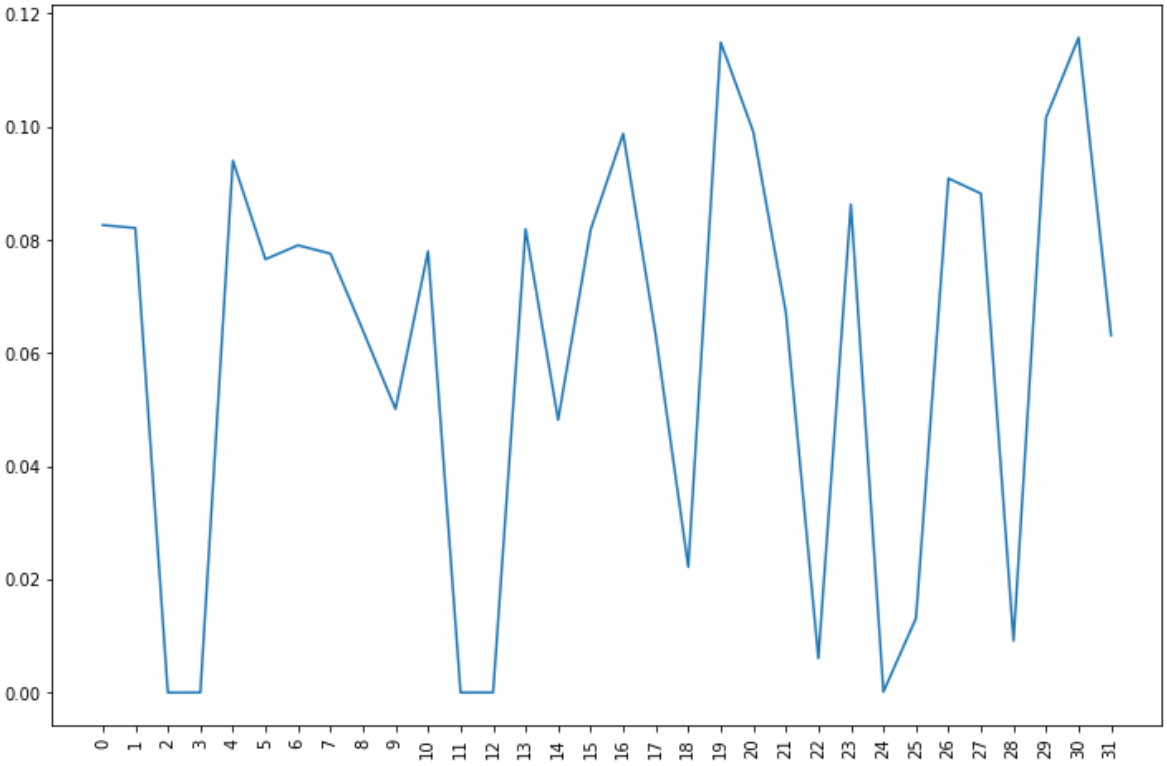
Activation for 1 kernel in 0 layer.

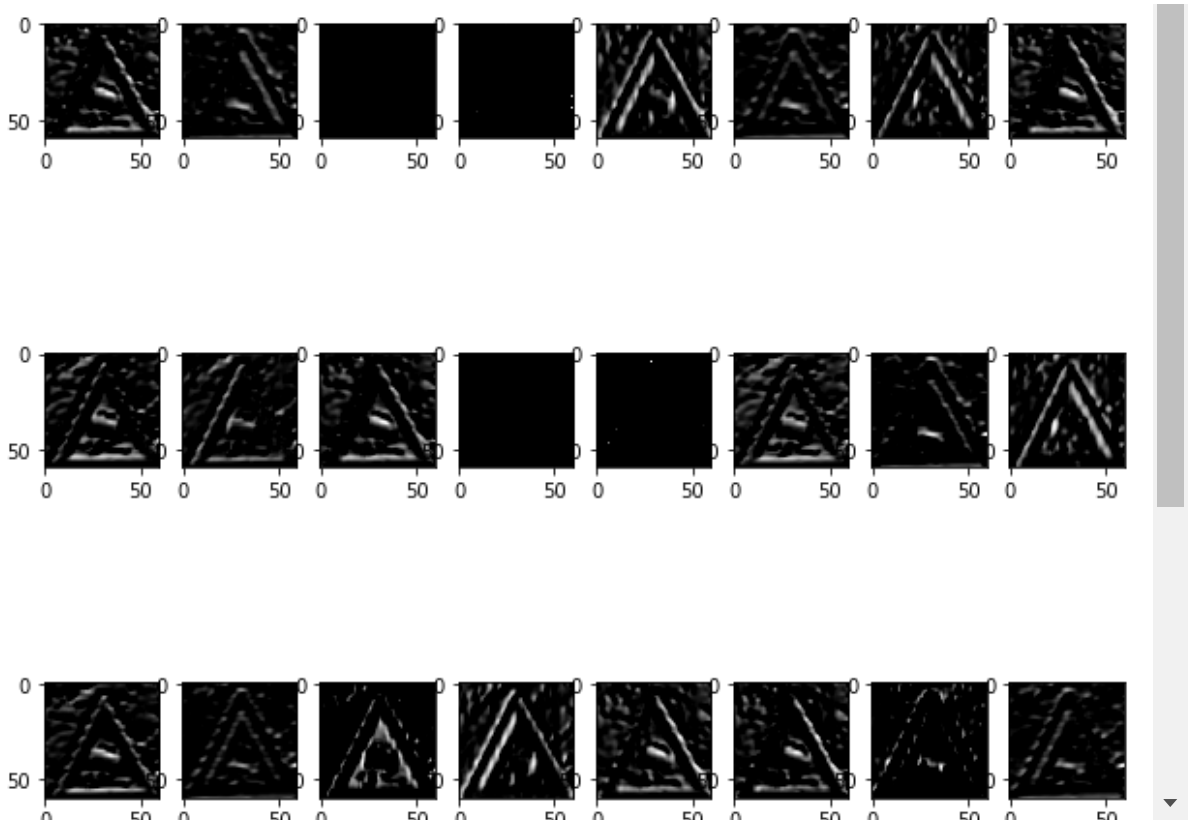


Probabilities of top 5 classes.
Actual class it belongs to: 23
Predicted class 30
Actual training image

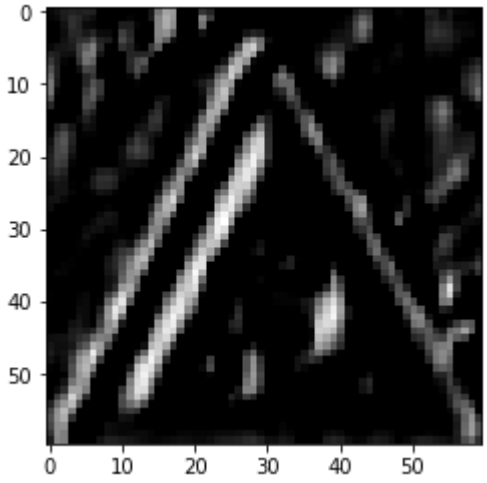



```
The kernel which activates/recognizes the shape: 30
```

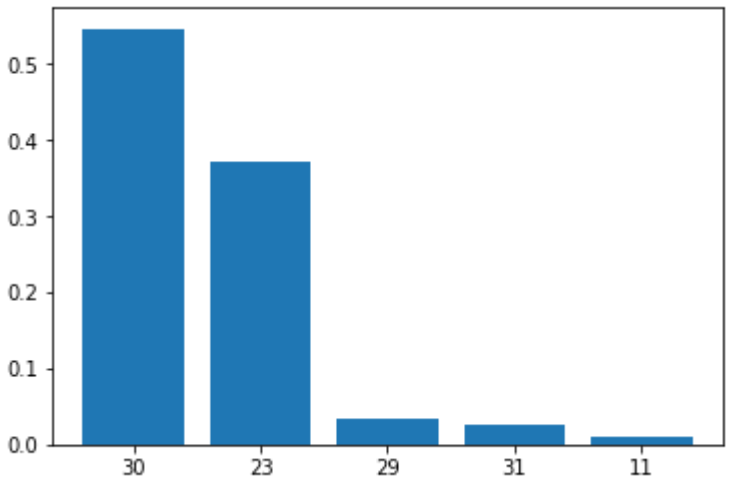




Multiple kernel activations starting from 0



Activation for 30 kernel in 1 layer.

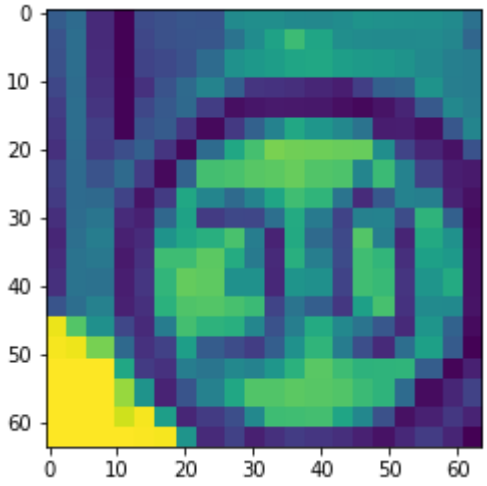


Probabilities of top 5 classes.

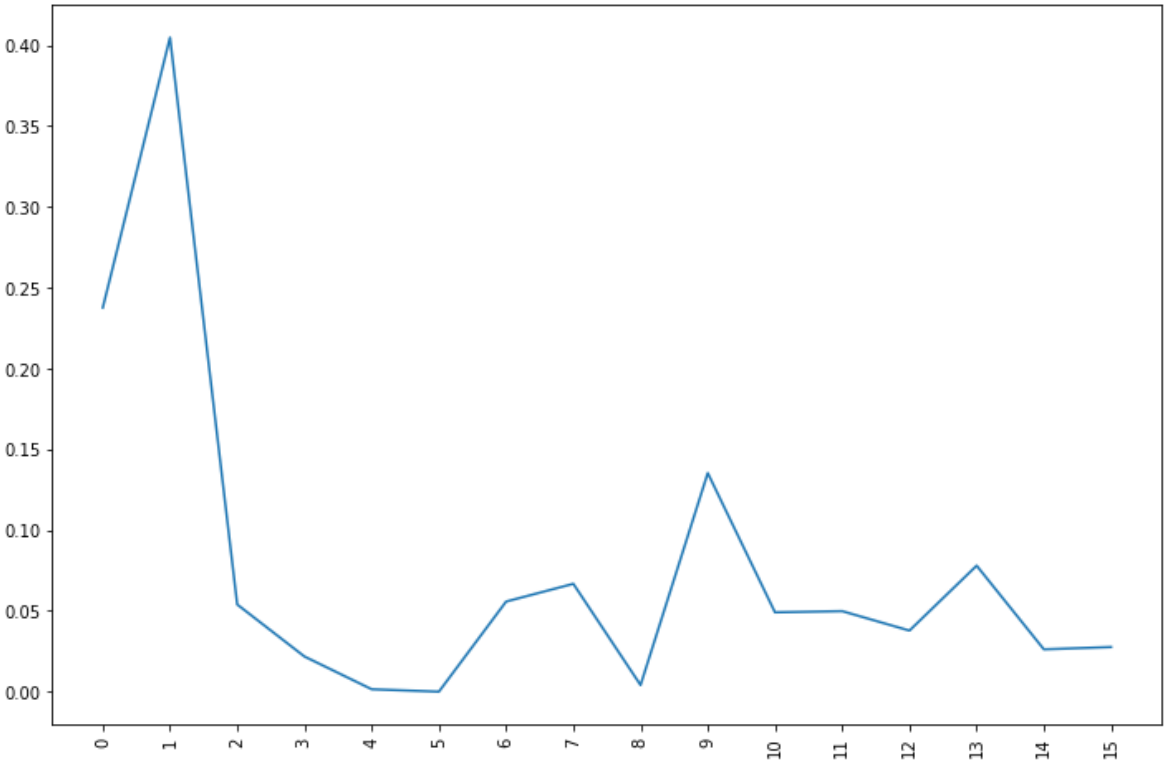
In [0]:

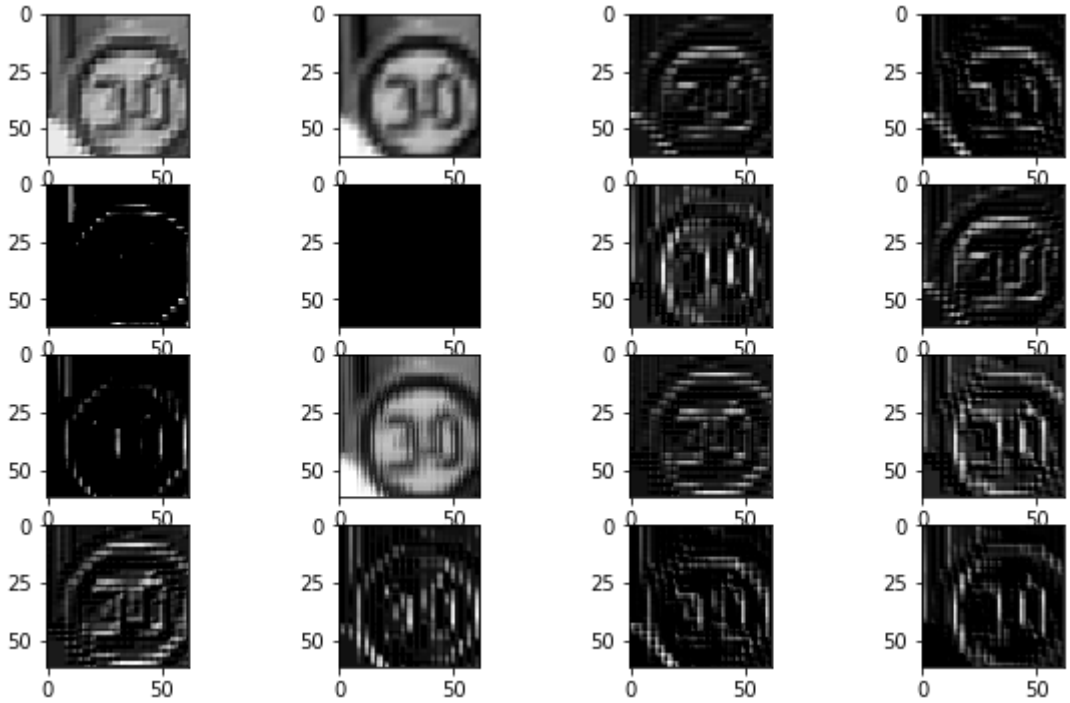
```
idx = 8627
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 4, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 1
Predicted class 2
Actual training image
```

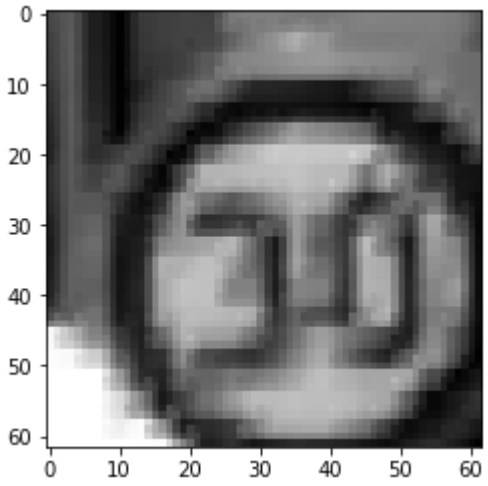



```
The kernel which activates/recognizes the shape: 1
```

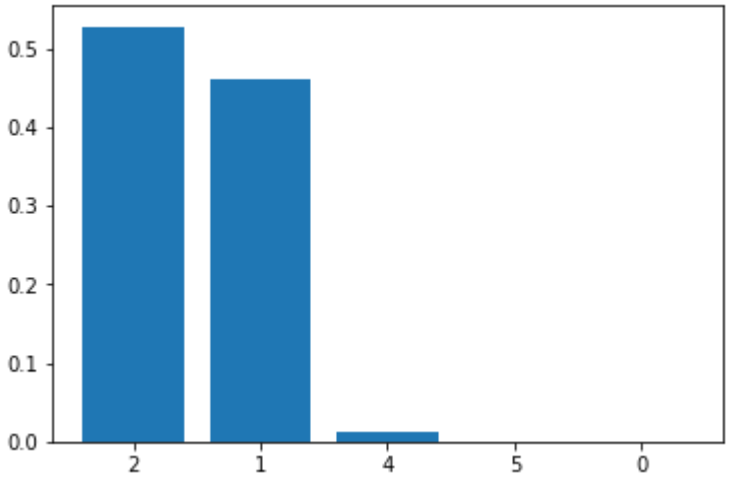




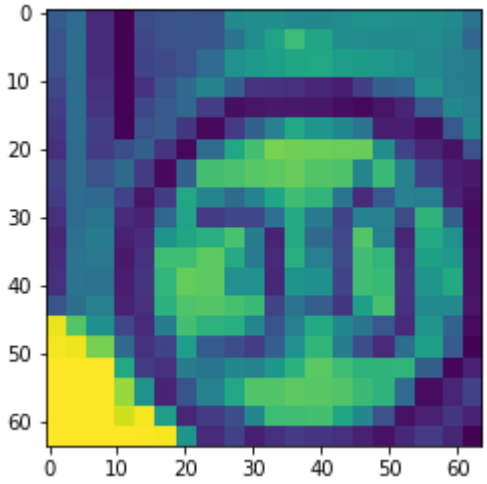
Multiple kernel activations starting from 0



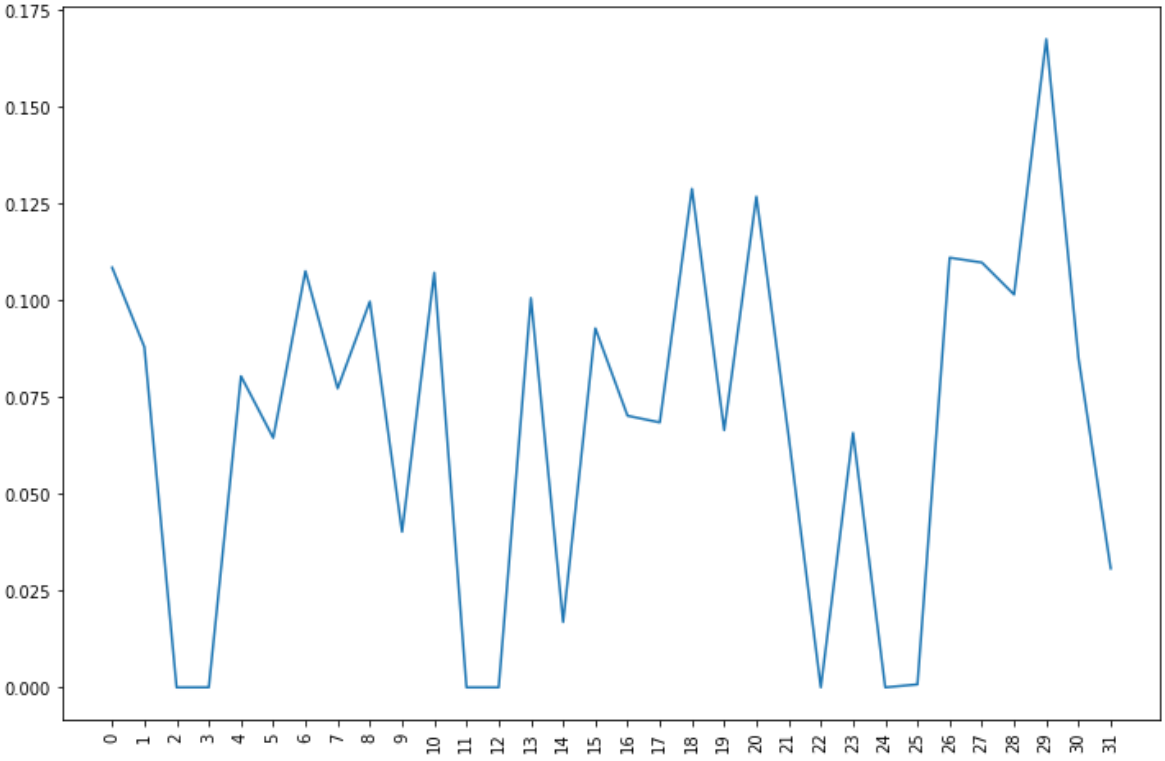
Activation for 1 kernel in 0 layer.

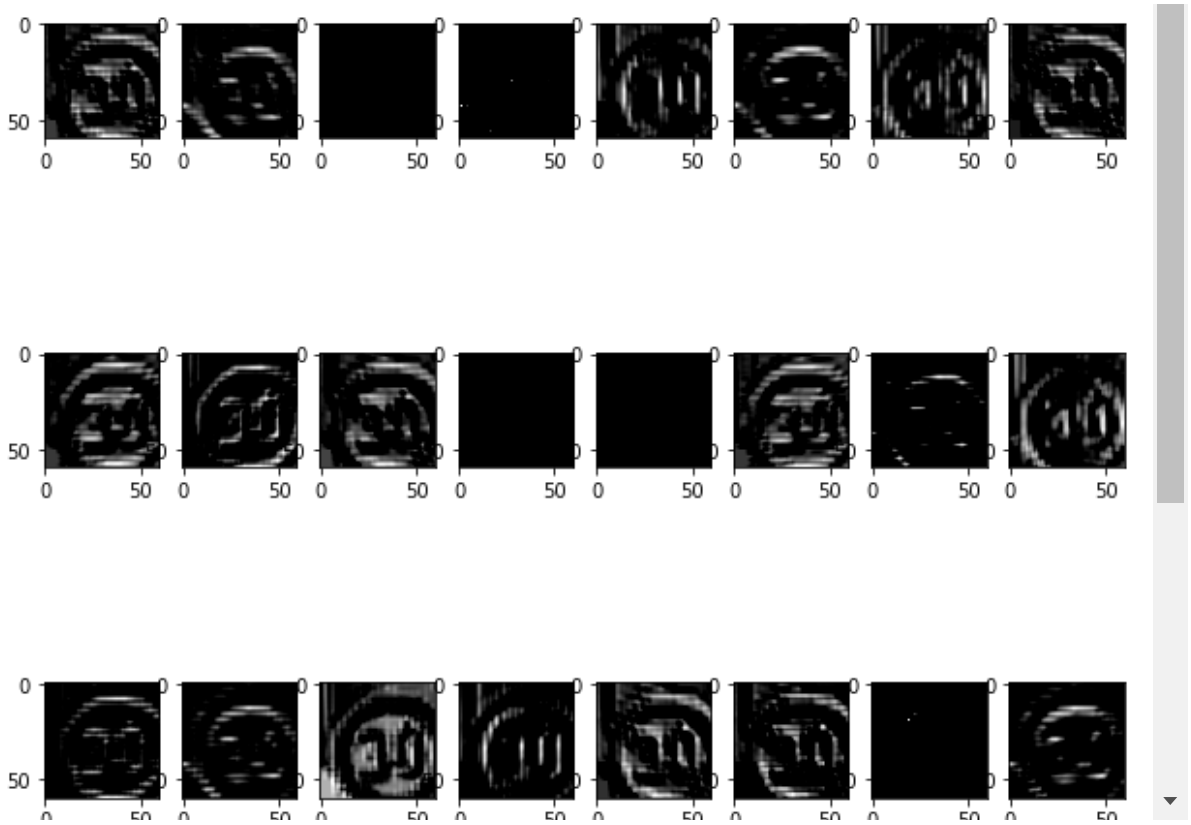


Probabilities of top 5 classes.
Actual class it belongs to: 1
Predicted class 2
Actual training image

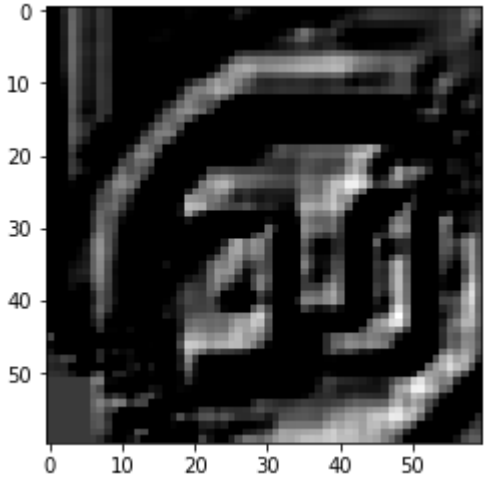



```
The kernel which activates/recognizes the shape: 29
```

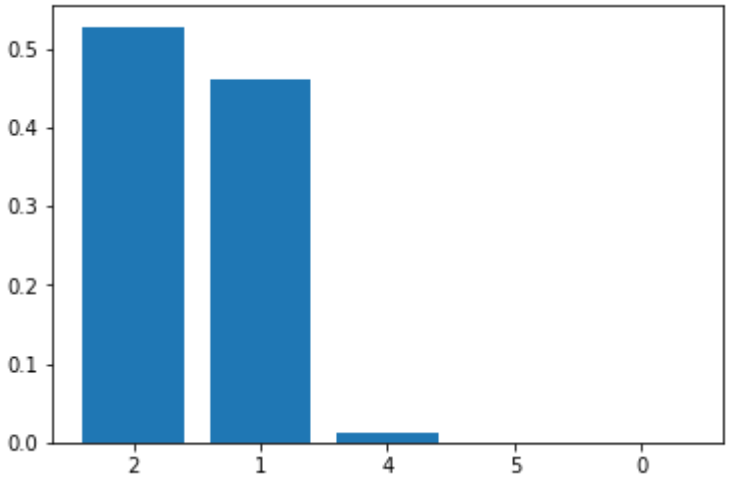




Multiple kernel activations starting from 0



Activation for 29 kernel in 1 layer.



Probabilities of top 5 classes.

[5.3] Conv(32 -- 4x4) - Conv(64 -- 3x3) - MaxPool(2x2) - Dropout(0.75) - Dense(128) - Dropout(0.5)

In [0]:

```
epochs = 15
model = Sequential()
model.add(Conv2D(
    32, kernel_size=(4, 4), activation='relu', input_shape=input_shape
))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.75))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(
    loss=keras.losses.categorical_crossentropy,
    optimizer=keras.optimizers.Adadelta(), metrics=['accuracy']
)

model.summary()

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 61, 61, 32)        544
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 59, 59, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 29, 29, 64)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 29, 29, 64)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 53824)             0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               6889600
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 43)                5547
=================================================================
Total params: 6,914,187
Trainable params: 6,914,187
Non-trainable params: 0
_________________________________________________________________
Train on 27446 samples, validate on 11763 samples
Epoch 1/15
27446/27446 [==============================] - 11s 393us/step - loss: 2.0004
- acc: 0.4785 - val_loss: 0.5220 - val_acc: 0.8592
Epoch 2/15
27446/27446 [==============================] - 9s 339us/step - loss: 0.5788
- acc: 0.8378 - val_loss: 0.2463 - val_acc: 0.9469
Epoch 3/15
27446/27446 [==============================] - 9s 342us/step - loss: 0.3590
- acc: 0.8985 - val_loss: 0.1501 - val_acc: 0.9652
Epoch 4/15
```

```
27446/27446 [==============================] - 9s 345us/step - loss: 0.2738
- acc: 0.9210 - val_loss: 0.1285 - val_acc: 0.9660
Epoch 5/15
27446/27446 [==============================] - 9s 345us/step - loss: 0.2267
- acc: 0.9353 - val_loss: 0.1153 - val_acc: 0.9724
Epoch 6/15
27446/27446 [==============================] - 10s 346us/step - loss: 0.1961
- acc: 0.9428 - val_loss: 0.1106 - val_acc: 0.9735
Epoch 7/15
27446/27446 [==============================] - 10s 346us/step - loss: 0.1767
- acc: 0.9478 - val_loss: 0.0885 - val_acc: 0.9793
Epoch 8/15
27446/27446 [==============================] - 9s 345us/step - loss: 0.1489
- acc: 0.9563 - val_loss: 0.0821 - val_acc: 0.9790
Epoch 9/15
27446/27446 [==============================] - 9s 343us/step - loss: 0.1406
- acc: 0.9589 - val_loss: 0.0778 - val_acc: 0.9821
Epoch 10/15
27446/27446 [==============================] - 9s 342us/step - loss: 0.1244
- acc: 0.9619 - val_loss: 0.0708 - val_acc: 0.9823
Epoch 11/15
27446/27446 [==============================] - 9s 342us/step - loss: 0.1136
- acc: 0.9650 - val_loss: 0.0662 - val_acc: 0.9838
Epoch 12/15
27446/27446 [==============================] - 9s 343us/step - loss: 0.1060
- acc: 0.9672 - val_loss: 0.0637 - val_acc: 0.9845
Epoch 13/15
27446/27446 [==============================] - 9s 345us/step - loss: 0.0950
- acc: 0.9706 - val_loss: 0.0682 - val_acc: 0.9844
Epoch 14/15
27446/27446 [==============================] - 9s 344us/step - loss: 0.0955
- acc: 0.9699 - val_loss: 0.0620 - val_acc: 0.9852
Epoch 15/15
27446/27446 [==============================] - 9s 343us/step - loss: 0.0862
- acc: 0.9739 - val_loss: 0.0585 - val_acc: 0.9869
```

In [0]:

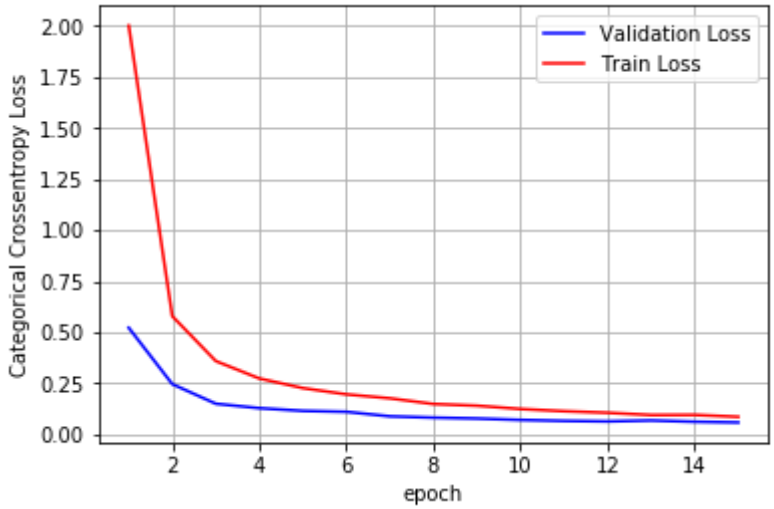
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(fig, x, vy, ty, ax)
```

```
Test score: 0.05854669301502624
Test accuracy: 0.9869081016747429
```



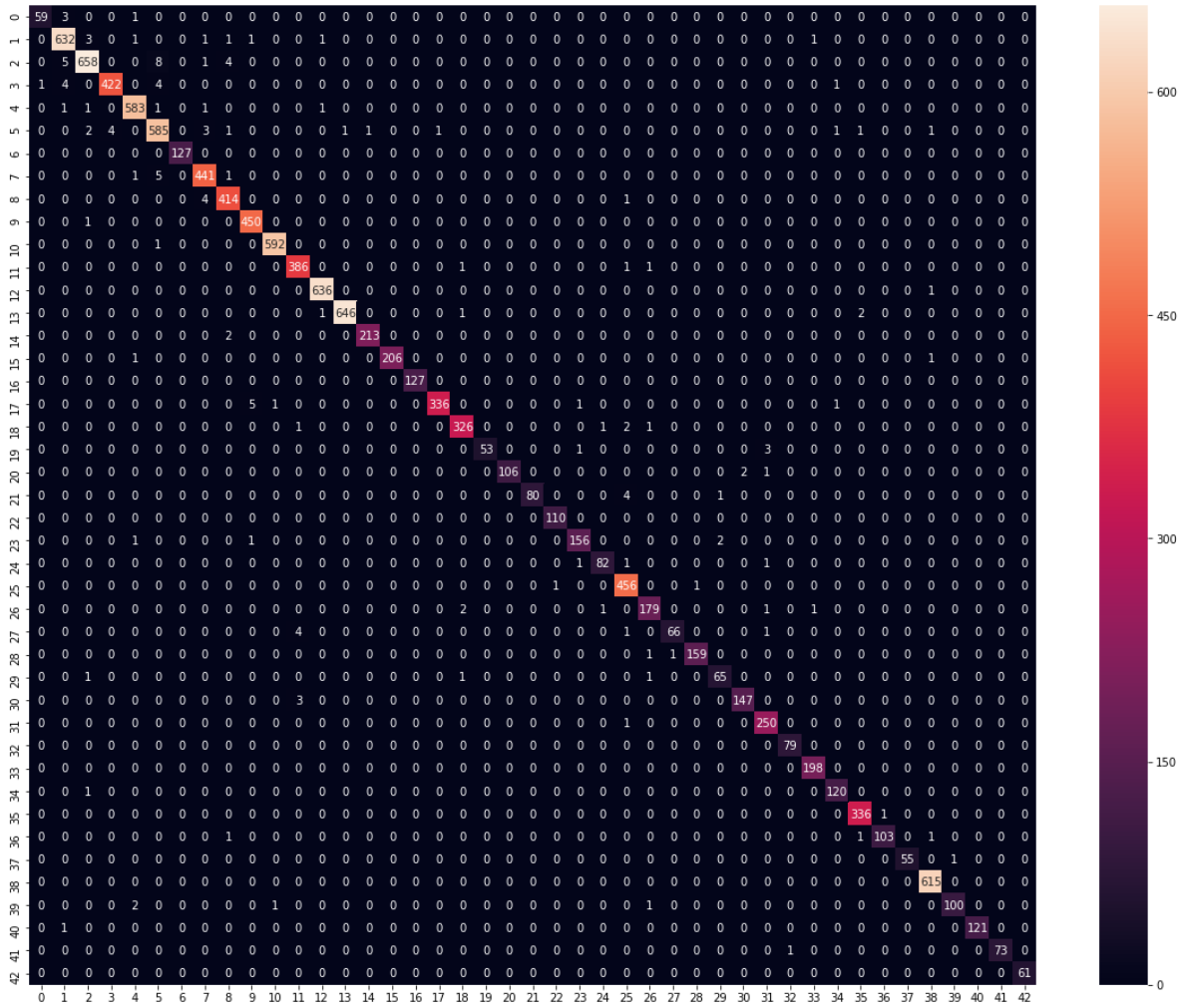
In [0]:

```
plt.figure(figsize=(20,16))
y_prediction = model.predict(x_test)
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_prediction, axis = 1)
# Convert validation observations to one hot vectors
y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
print("-------------------------------------------------------")
print("Micro F1 score", f1_score(y_true, y_pred_classes, average='micro'))
print("-------------------------------------------------------")
confusion_mtx = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(confusion_mtx, annot=True, fmt="d")
```

```
-------------------------------------------------------
Micro F1 score 0.9869081016747429
-------------------------------------------------------
```

Out[74]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b665ce390>
```



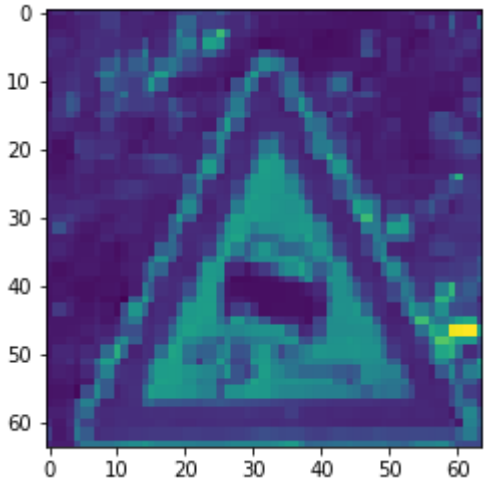
In [0]:

```
layer_outputs = [layer.output for layer in model.layers]
activation_model = Model(inputs=model.input, outputs=layer_outputs)
```

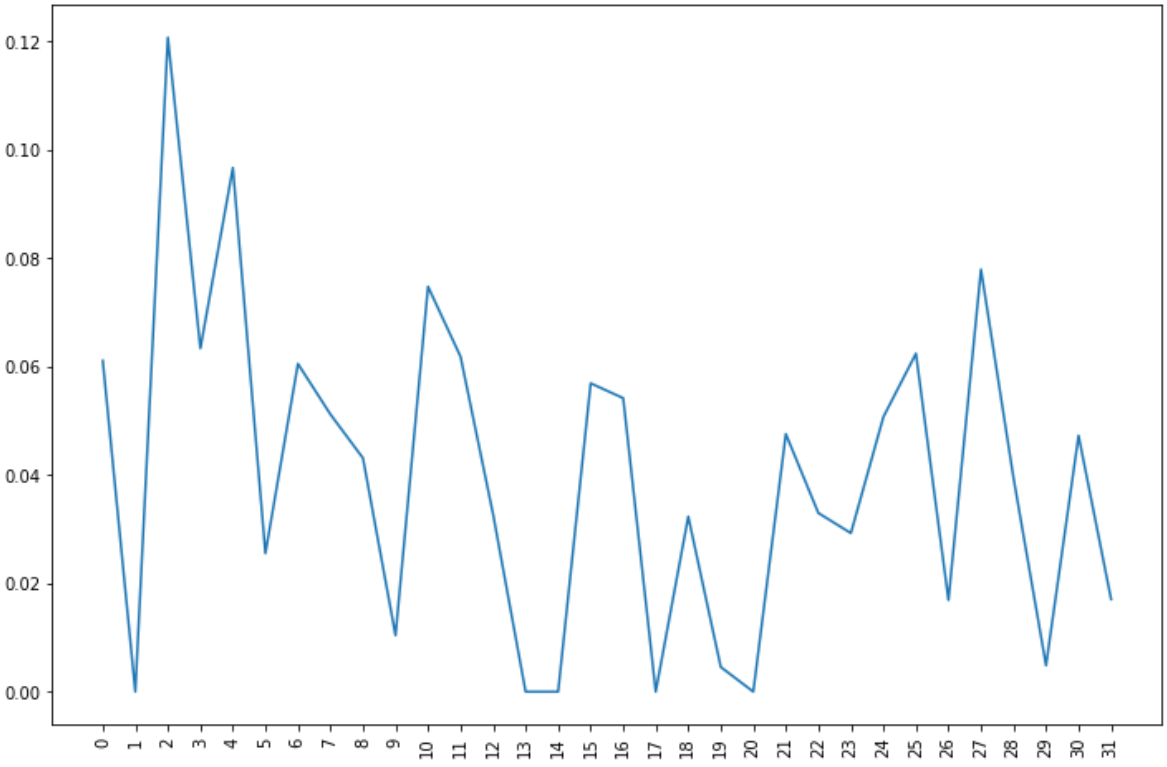
In [0]:

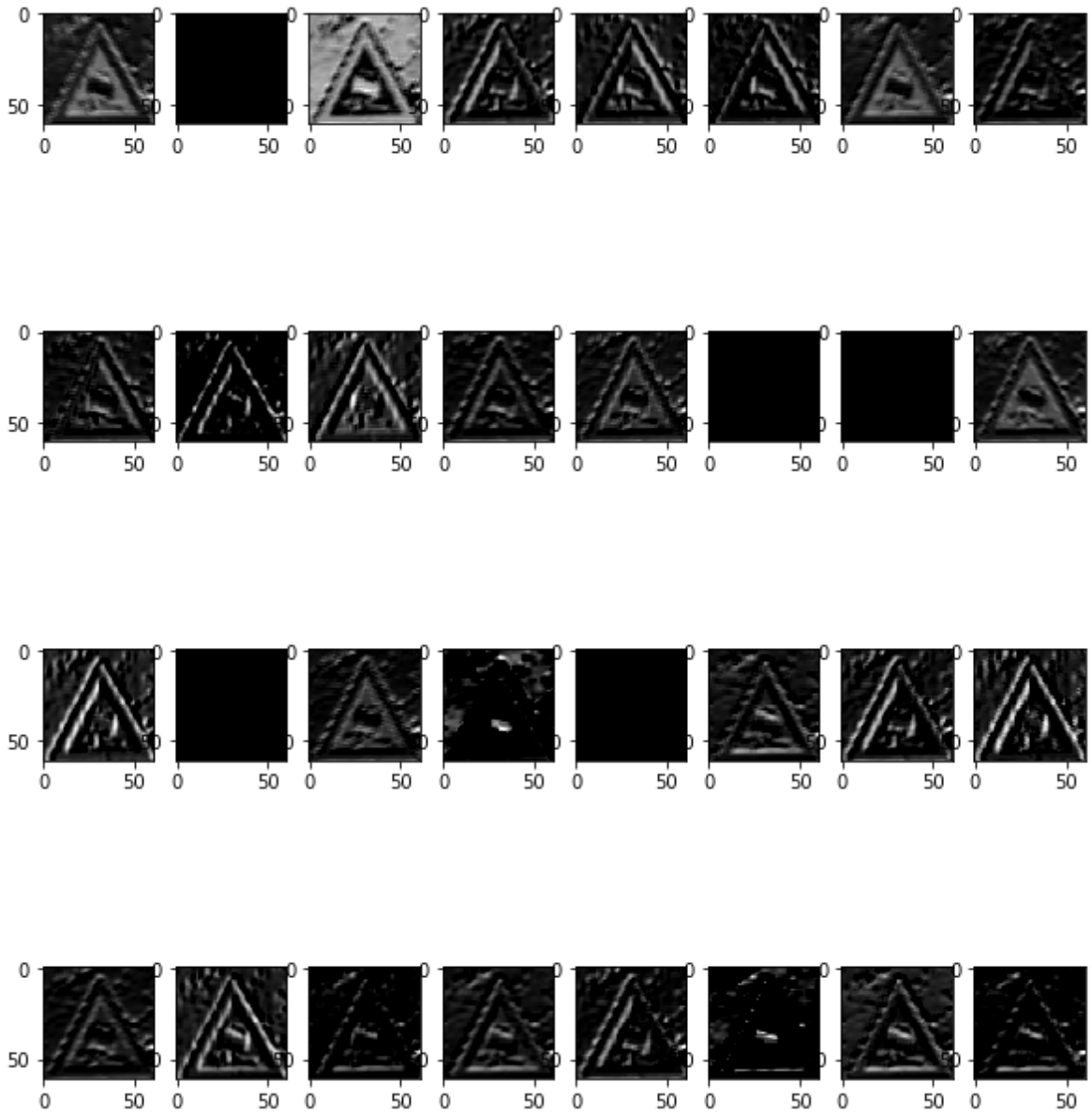
```
idx = 711
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 23
Predicted class 23
Actual training image
```

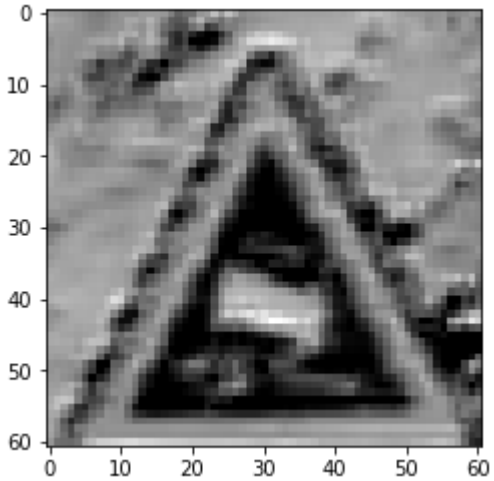



```
The kernel which activates/recognizes the shape: 2
```

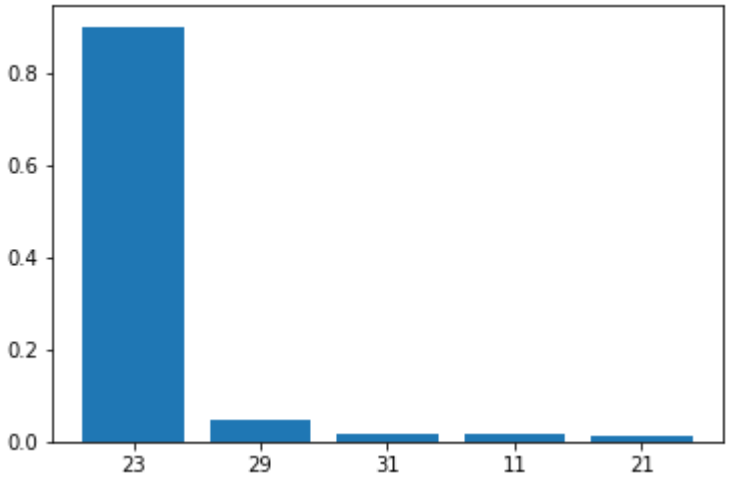




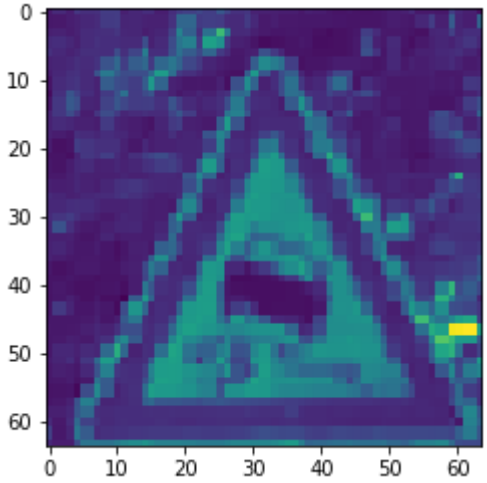
Multiple kernel activations starting from 0



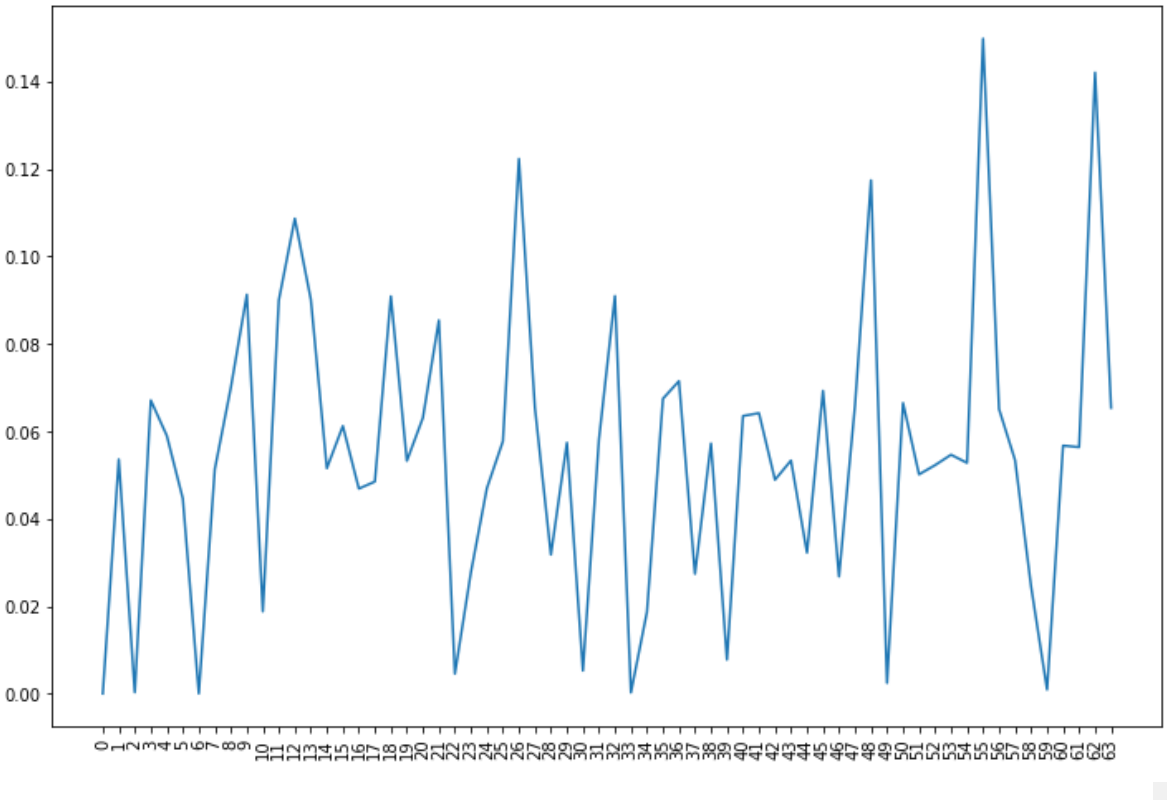
Activation for 2 kernel in 0 layer.

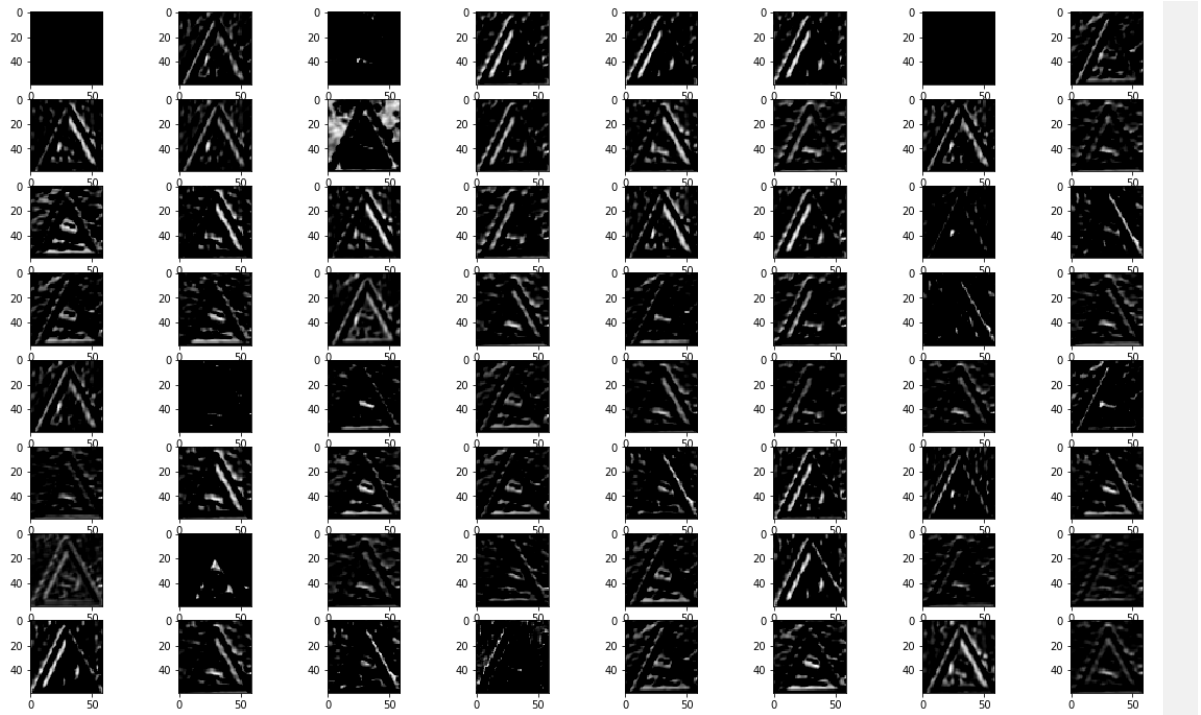



```
Probabilities of top 5 classes.
Actual class it belongs to: 23
Predicted class 23
Actual training image
```

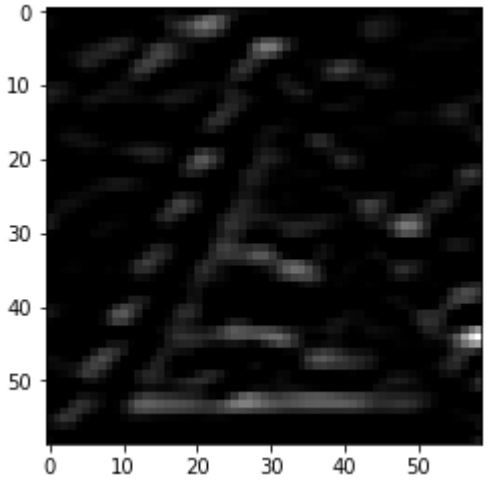



```
The kernel which activates/recognizes the shape: 55
```

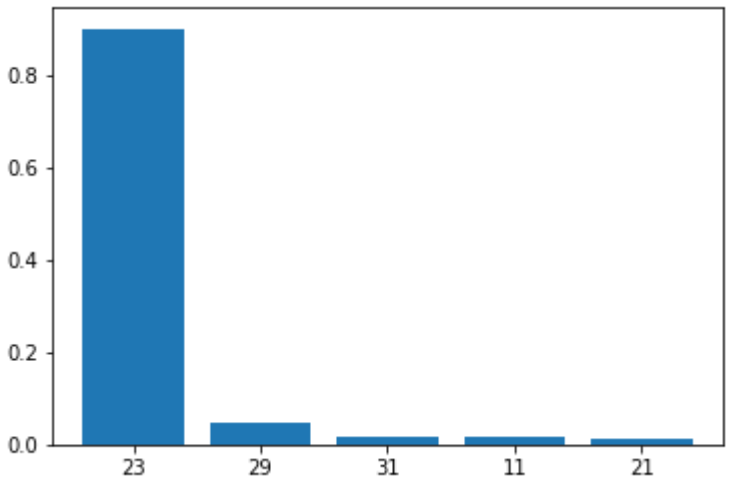




Multiple kernel activations starting from 0



Activation for 55 kernel in 1 layer.

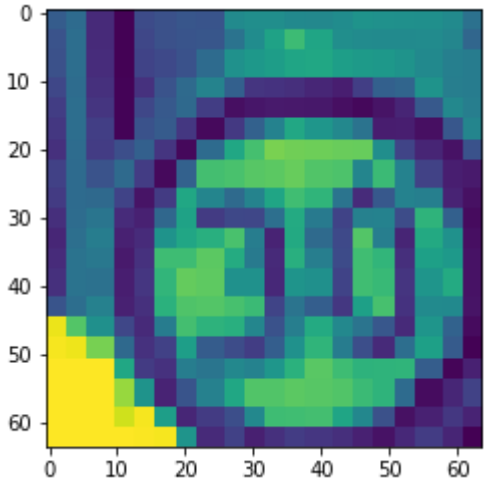


```
Probabilities of top 5 classes.
```

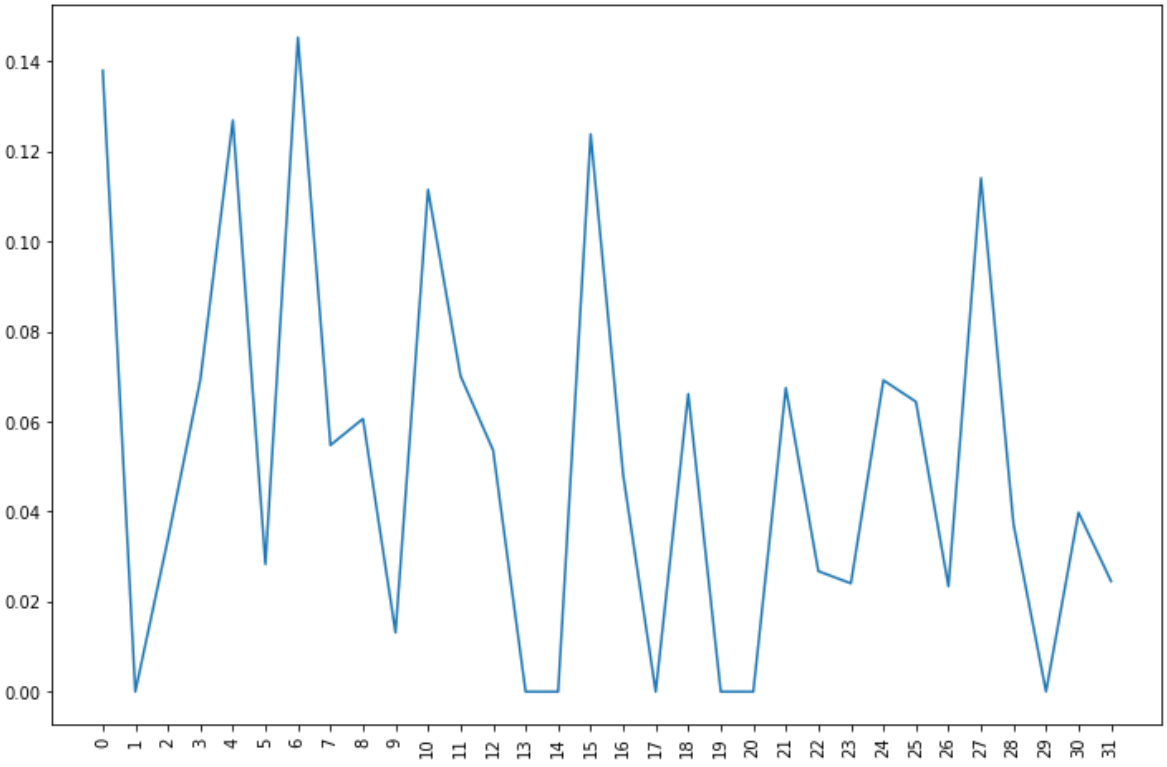
In [0]:

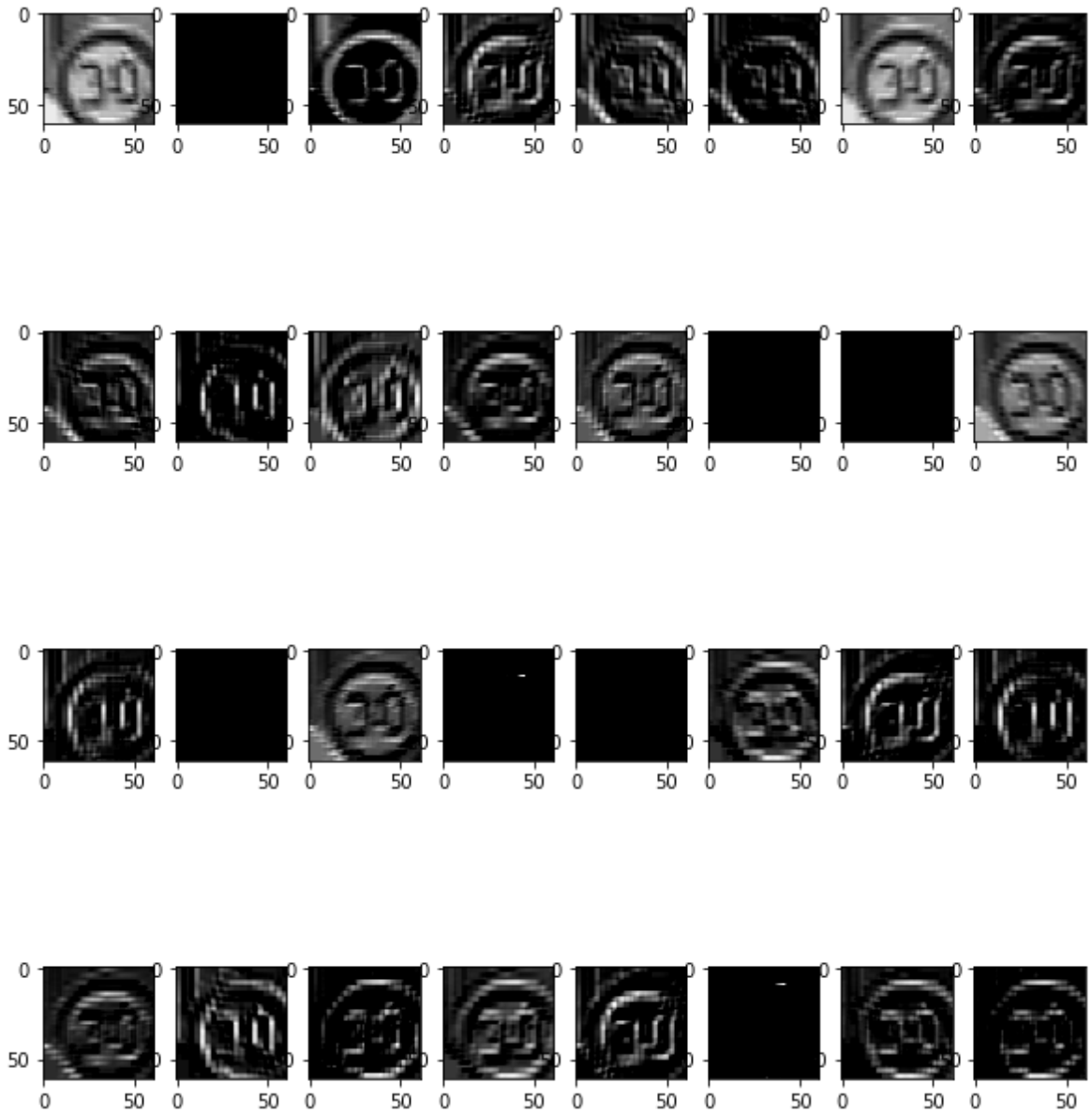
```
idx = 8627
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
```

```
Actual class it belongs to: 1
Predicted class 1
Actual training image
```

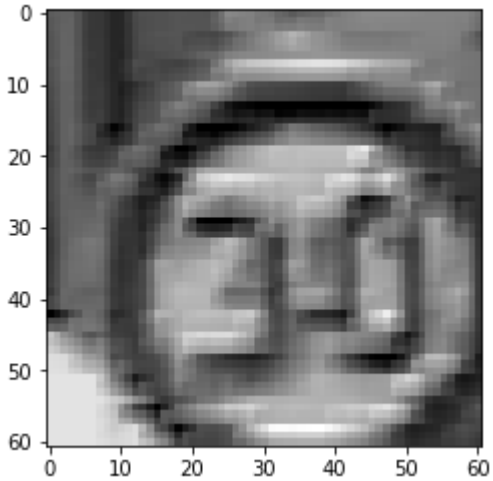



```
The kernel which activates/recognizes the shape: 6
```

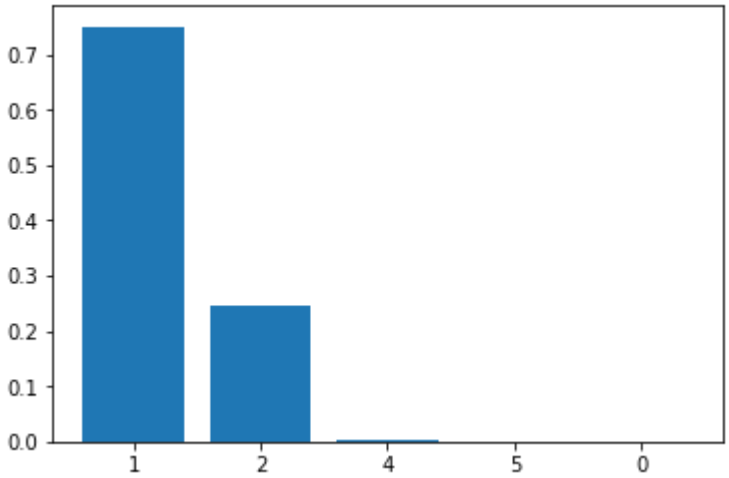




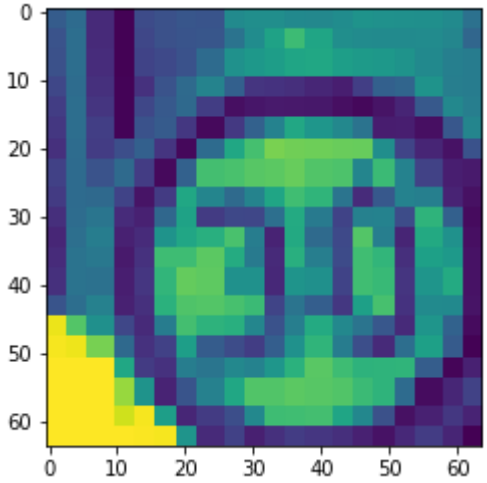
Multiple kernel activations starting from 0



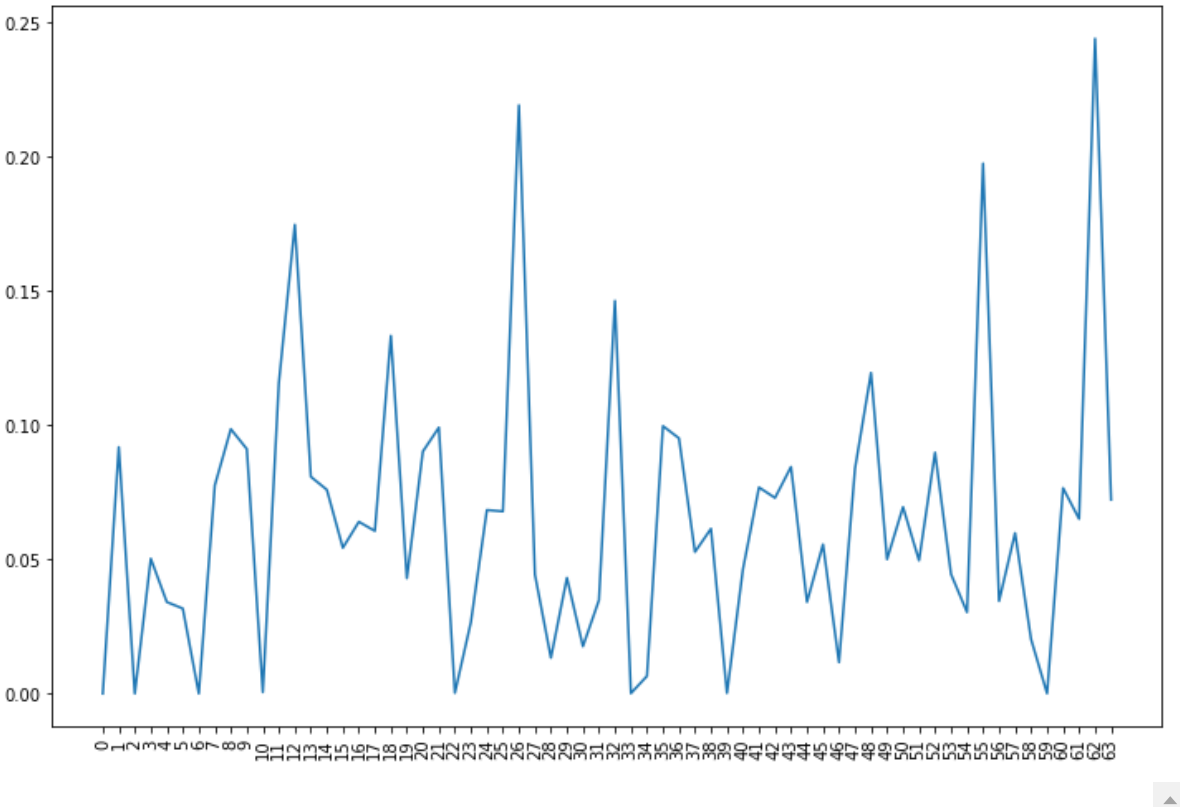
Activation for 6 kernel in 0 layer.

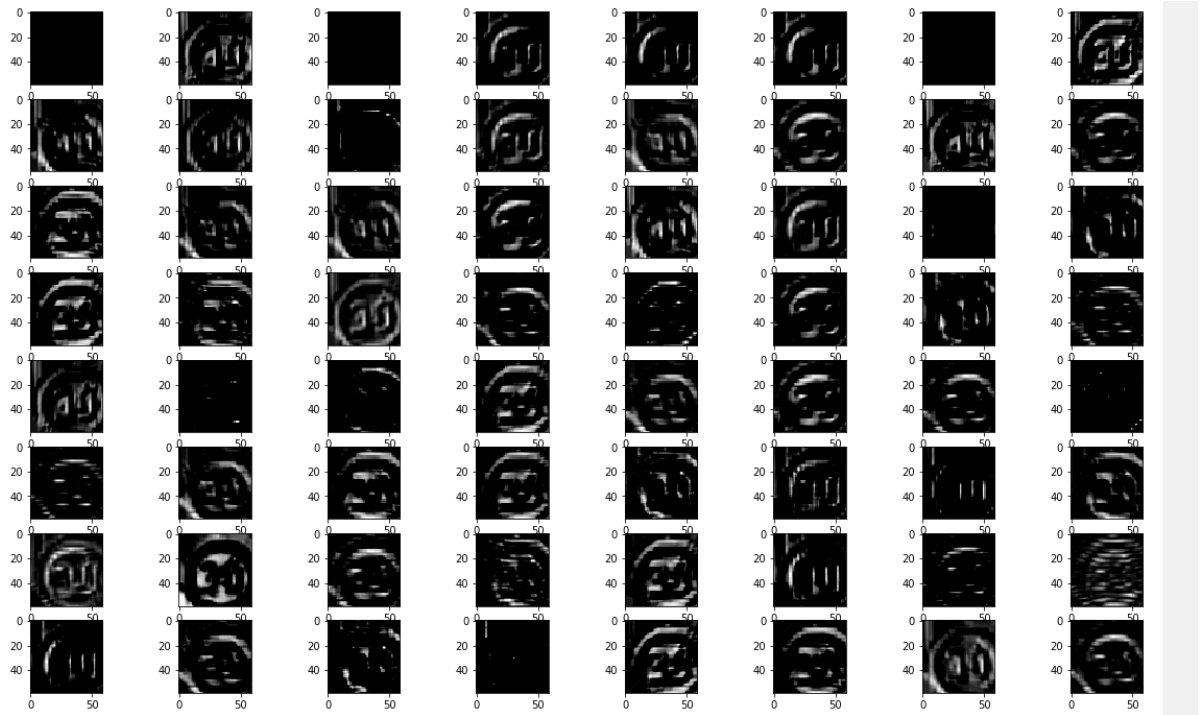



```
Probabilities of top 5 classes.
Actual class it belongs to: 1
Predicted class 1
Actual training image
```

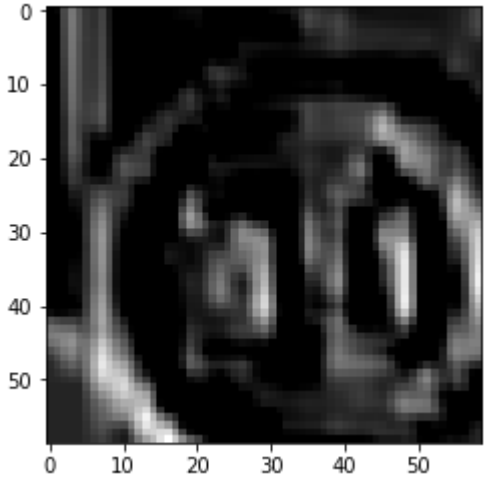



```
The kernel which activates/recognizes the shape: 62
```

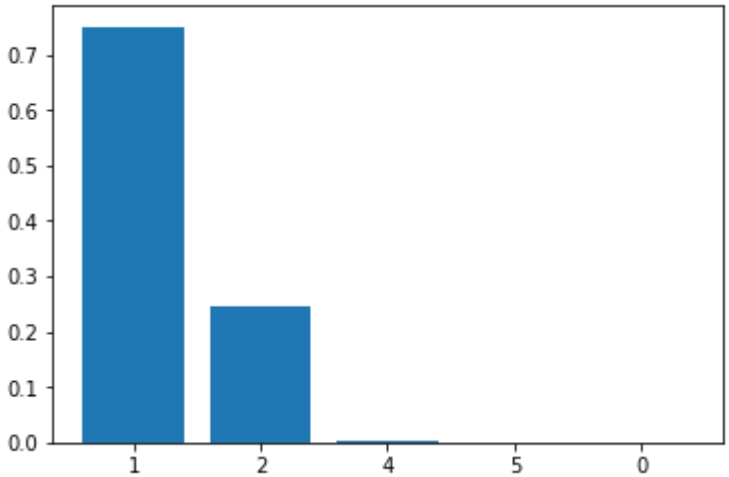




Multiple kernel activations starting from 0



Activation for 62 kernel in 1 layer.



Probabilities of top 5 classes.

In [0]:

```
actual_class = []
index_image = []
predicted_class = []
for idx, im in enumerate(y_true):
    if im != np.argmax(y_prediction[idx]):
        actual_class.append(im)
        predicted_class.append(np.argmax(y_prediction[idx]))
        index_image.append(idx)
np.array(index_image)
```

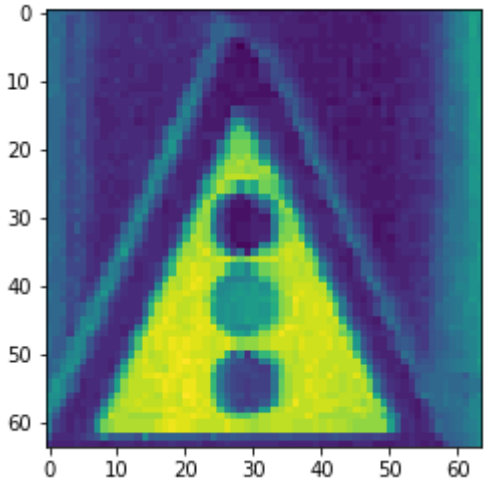
Out[78]:

```
array([   20,    37,    41,   151,   157,   163,   569,   592,   621,
         684,   696,   803,   836,   981,  1097,  1188,  1193,  1210,
        1321,  1391,  1404,  1478,  1609,  1737,  1748,  1750,  1840,
        1880,  1915,  1926,  1993,  1998,  2325,  2361,  2365,  2722,
        2791,  2873,  2967,  2978,  3055,  3080,  3113,  3450,  3489,
        3509,  3510,  3513,  3559,  3576,  3592,  3629,  3665,  3669,
        3785,  3846,  3853,  3953,  4083,  4101,  4123,  4302,  4316,
        4375,  4558,  4921,  4926,  5082,  5089,  5240,  5446,  5470,
        5491,  5513,  5802,  5855,  6017,  6102,  6157,  6166,  6170,
        6175,  6384,  6669,  6795,  6806,  6867,  6930,  6985,  6995,
        7269,  7295,  7358,  7377,  7378,  7481,  7496,  7520,  7521,
        7570,  7626,  7700,  7745,  7796,  7871,  7968,  8047,  8200,
        8207,  8235,  8346,  8348,  8412,  8492,  8510,  8578,  8598,
        8632,  8677,  8723,  8735,  8739,  8876,  9477,  9584,  9749,
        9892, 10073, 10098, 10124, 10125, 10554, 10589, 10621, 10917,
       10969, 10972, 11042, 11078, 11090, 11120, 11178, 11211, 11301,
       11307, 11366, 11404, 11483, 11527, 11601, 11653, 11684, 11697,
       11711])
```

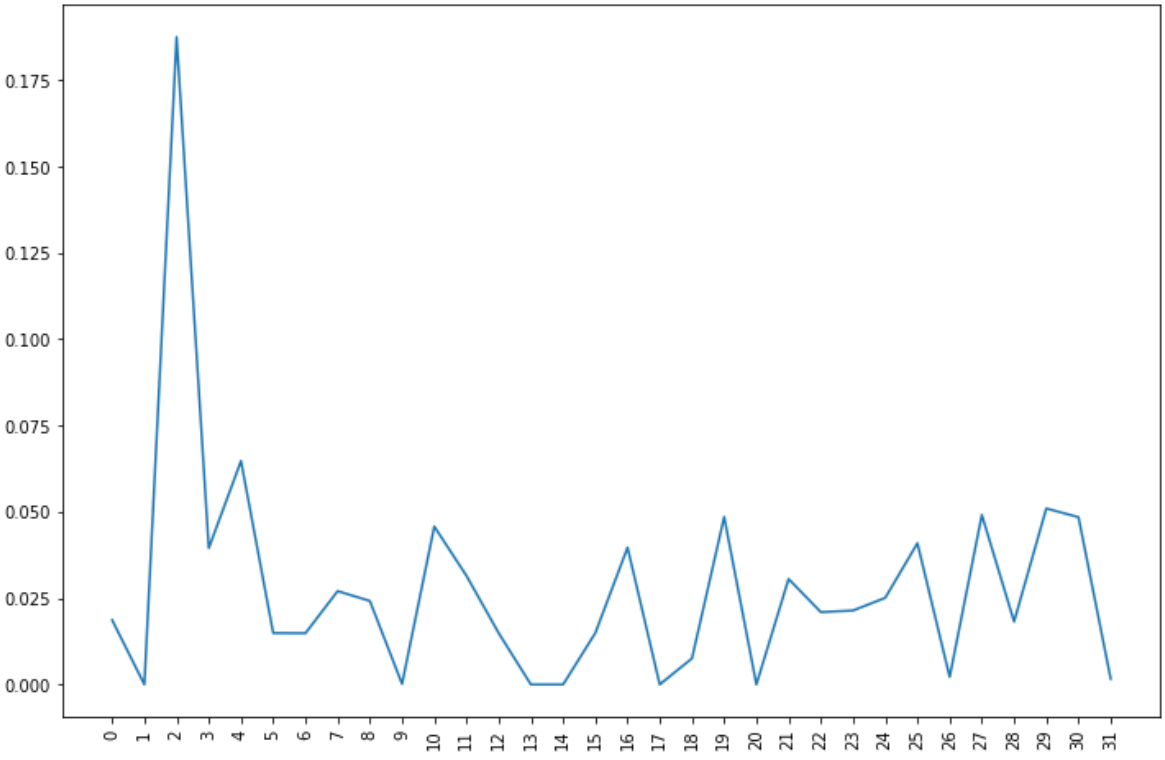
In [0]:

```
idx = 684
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
```

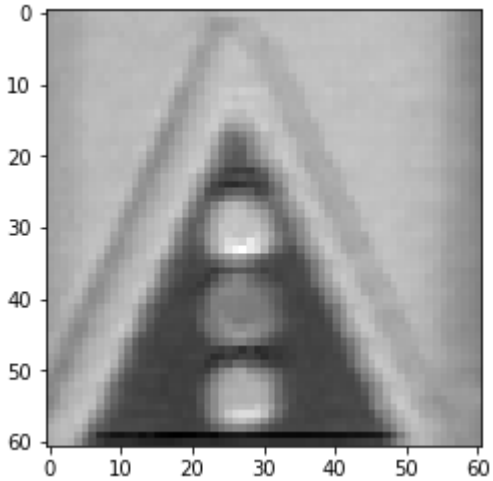
```
Actual class it belongs to: 26
Predicted class 31
Actual training image
```



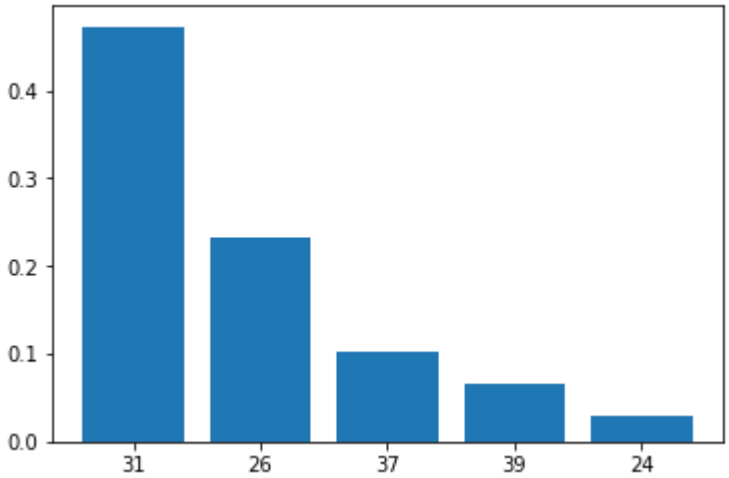

```
The kernel which activates/recognizes the shape: 2
```



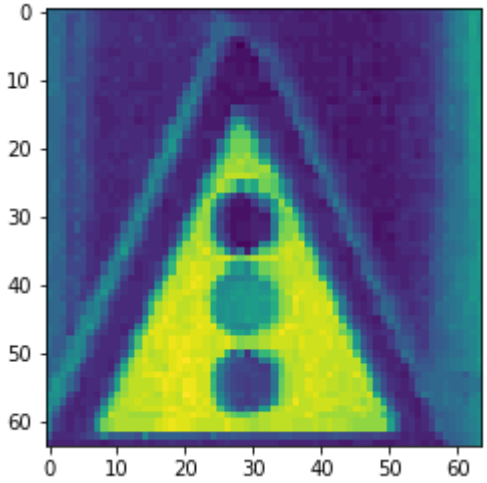
Multiple kernel activations starting from 0



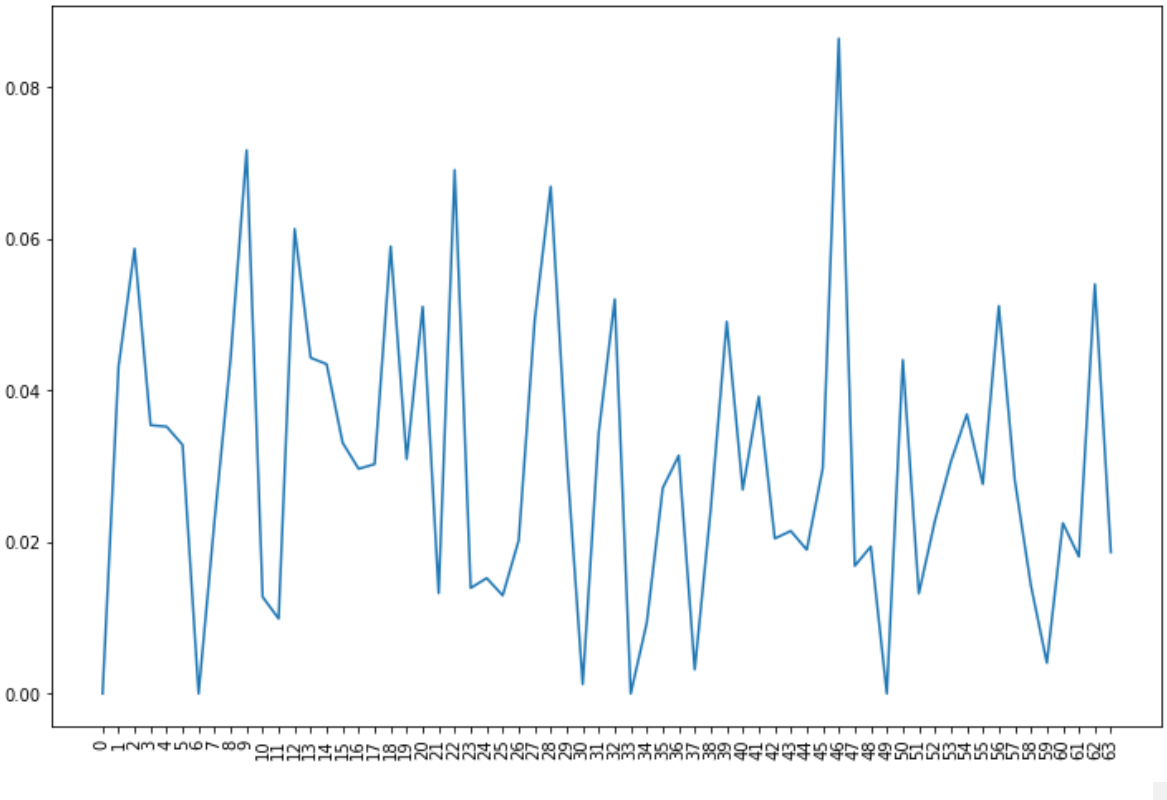
Activation for 2 kernel in 0 layer.

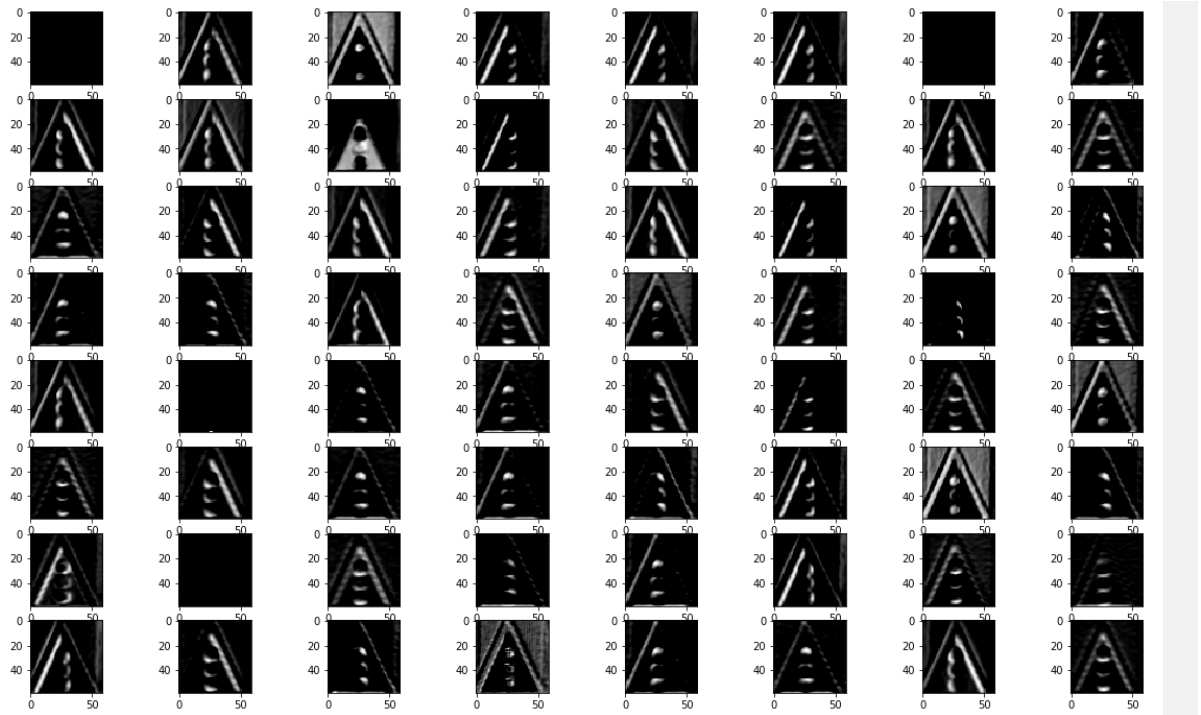



```
Probabilities of top 5 classes.
Actual class it belongs to: 26
Predicted class 31
Actual training image
```

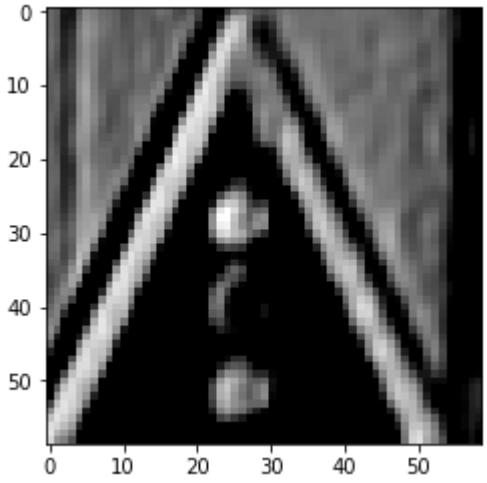



```
The kernel which activates/recognizes the shape: 46
```

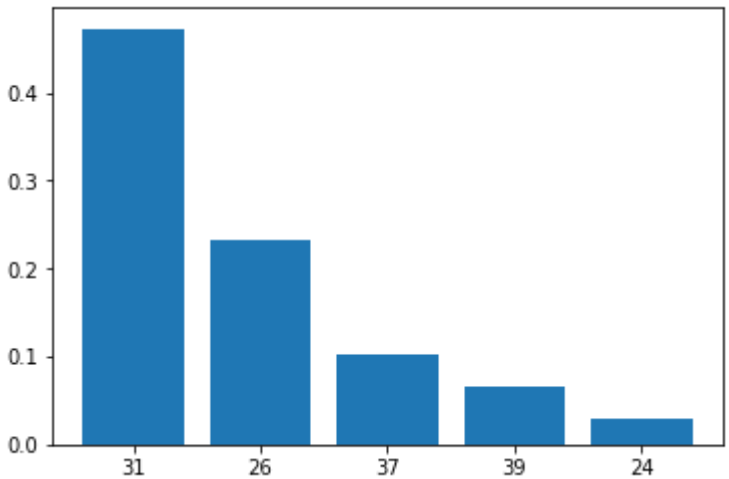




Multiple kernel activations starting from 0



Activation for 46 kernel in 1 layer.

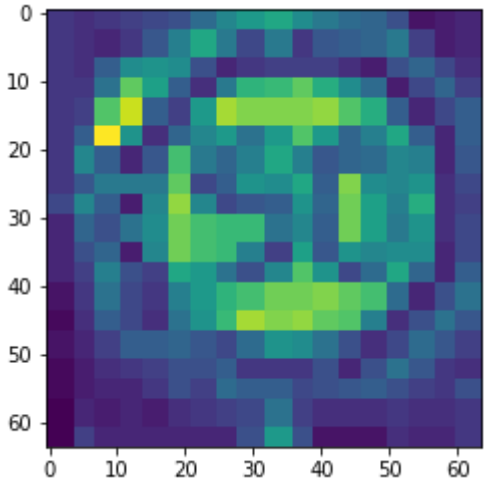


```
Probabilities of top 5 classes.
```

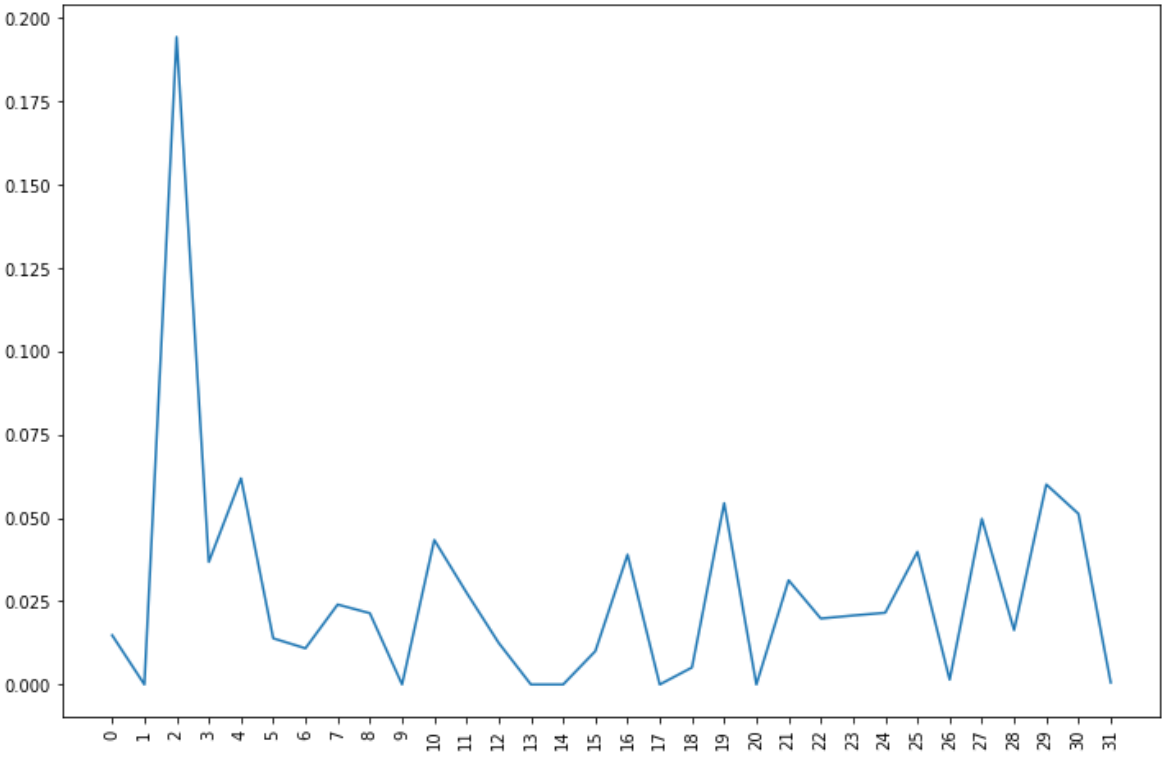
In [0]:

```
idx = 11078
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
```

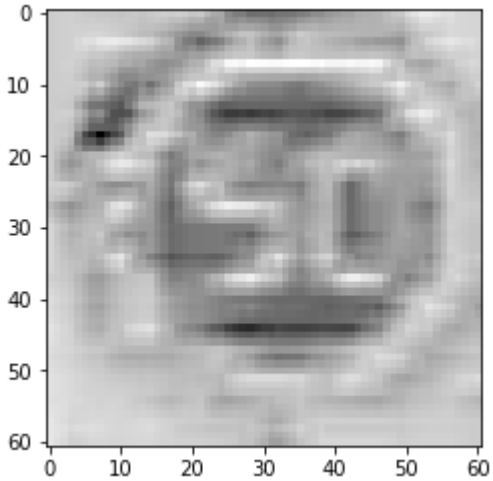
```
Actual class it belongs to: 2
Predicted class 8
Actual training image
```



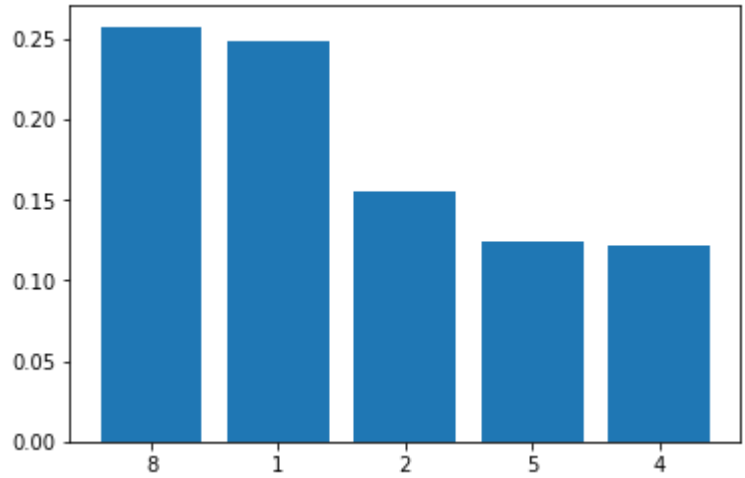

```
The kernel which activates/recognizes the shape: 2
```



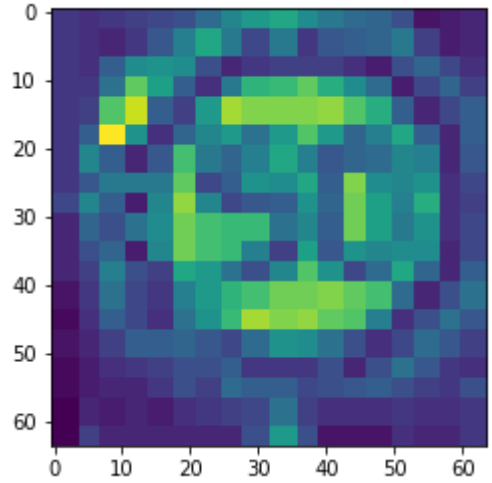
Multiple kernel activations starting from 0



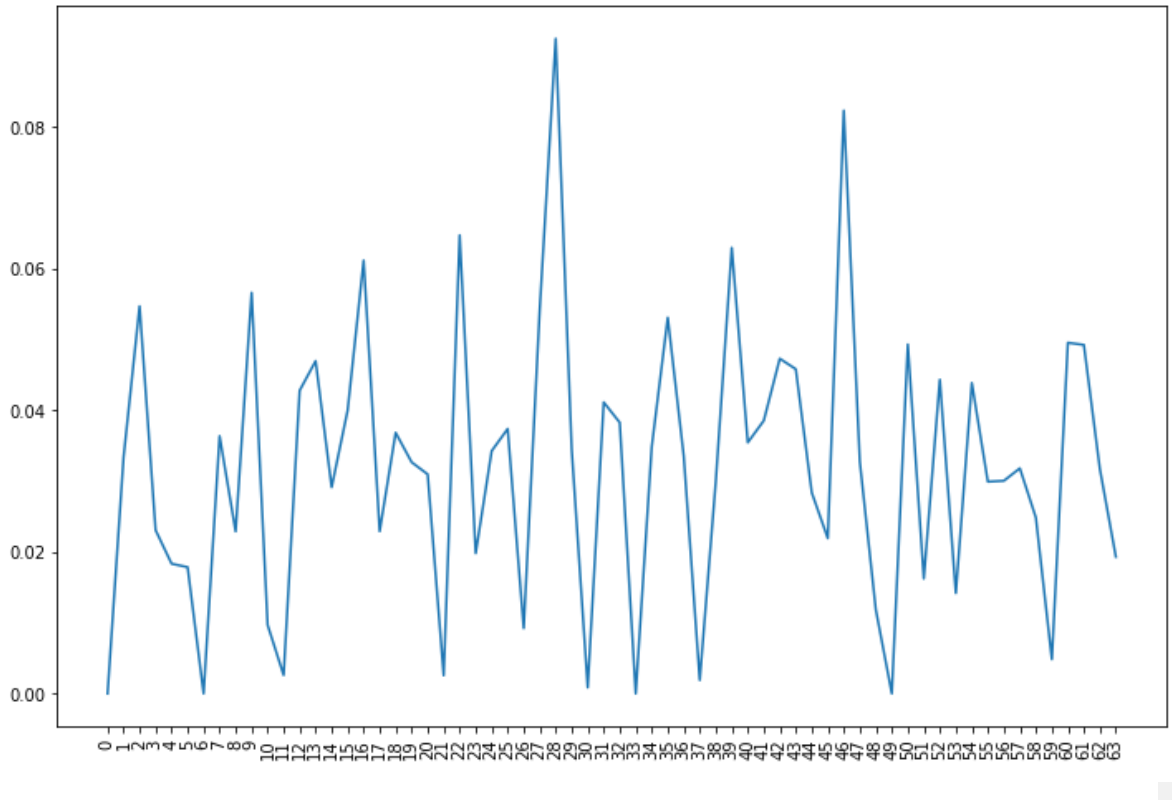
Activation for 2 kernel in 0 layer.

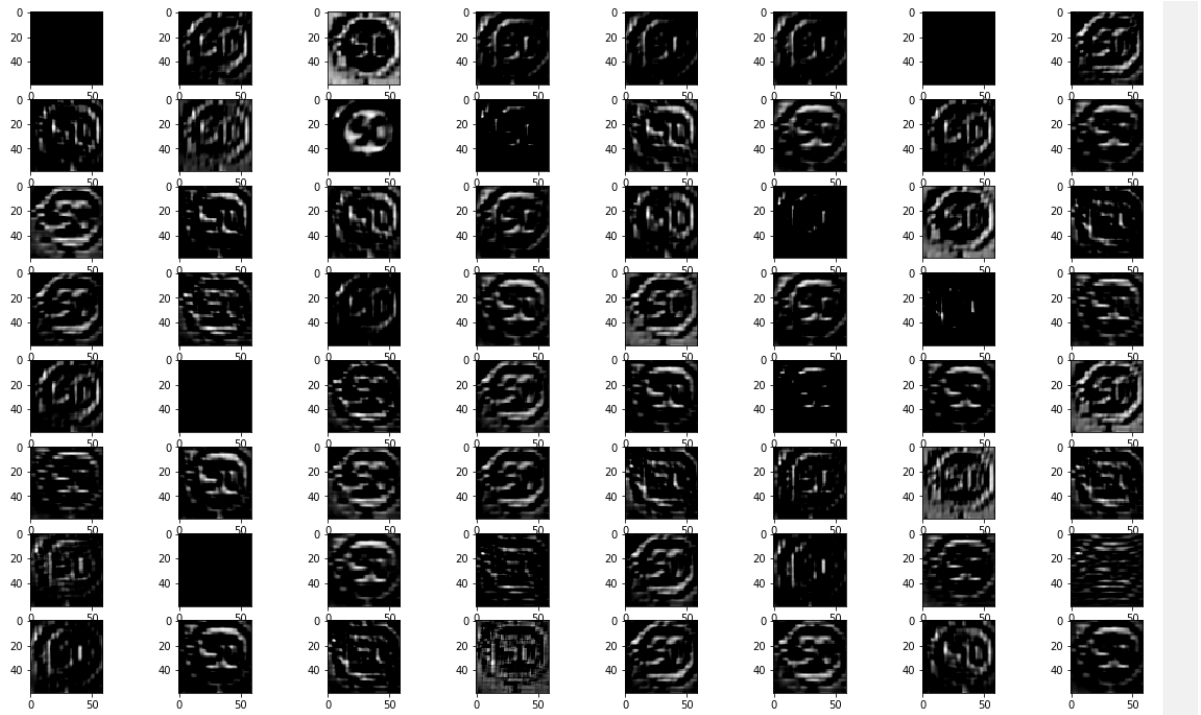



```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 8
Actual training image
```

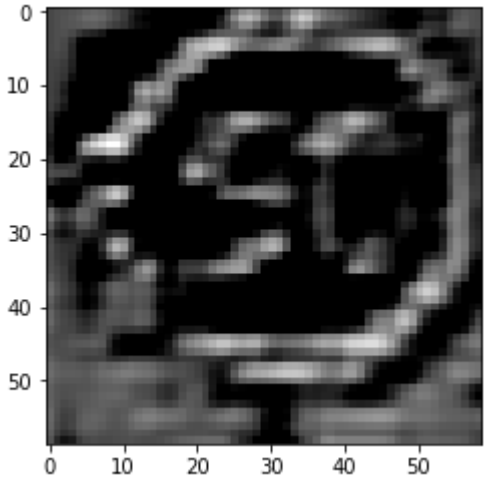



```
The kernel which activates/recognizes the shape: 28
```

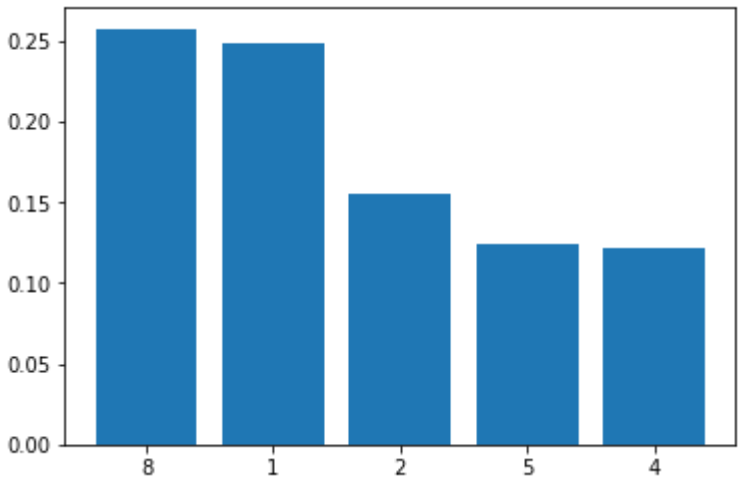




Multiple kernel activations starting from 0



Activation for 28 kernel in 1 layer.

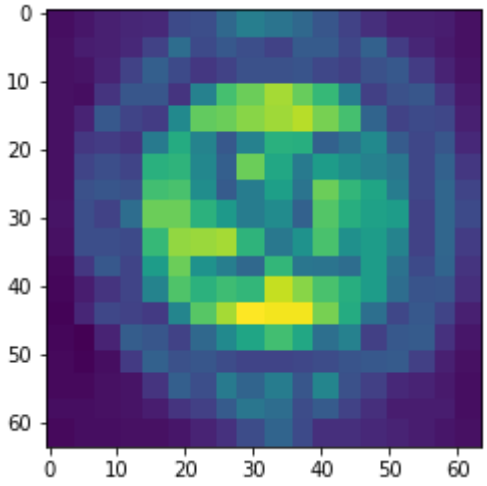


Probabilities of top 5 classes.

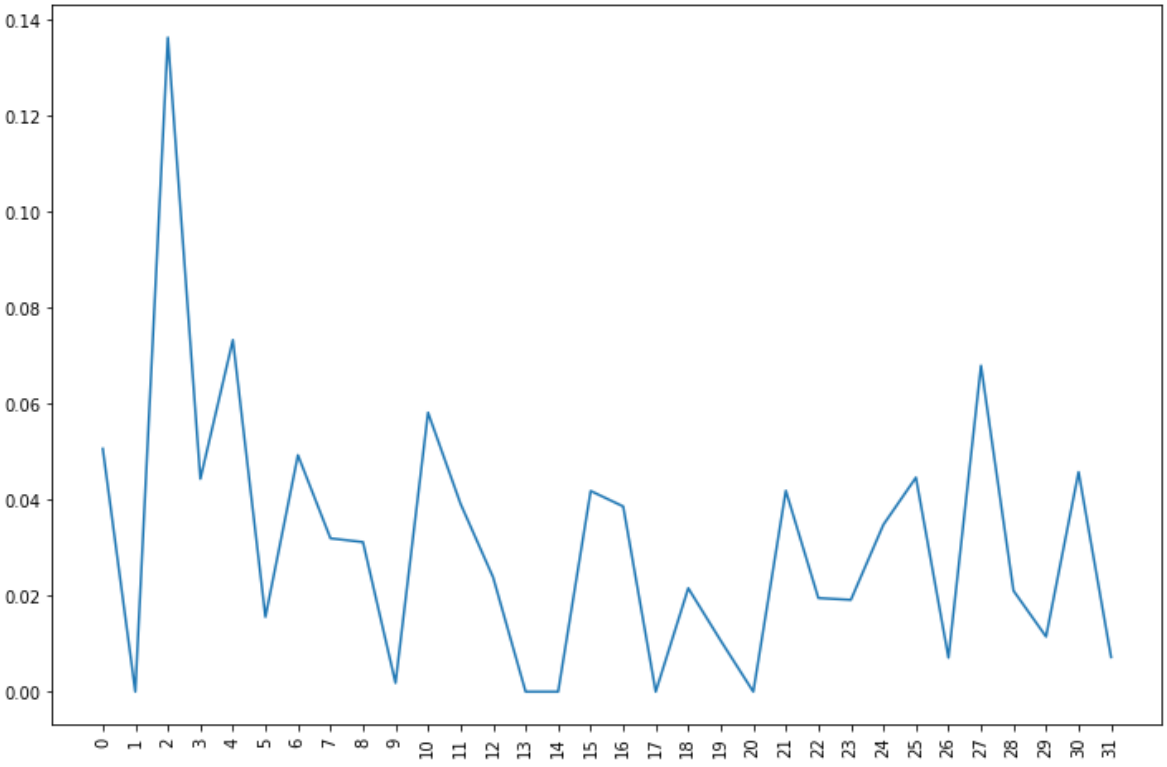
In [0]:

```
idx = 11527
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
```

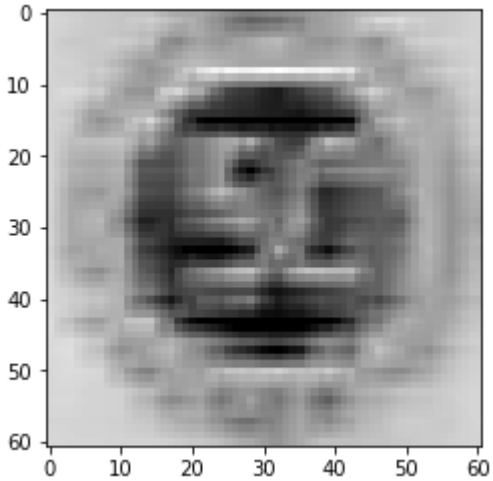
```
Actual class it belongs to: 2
Predicted class 1
Actual training image
```



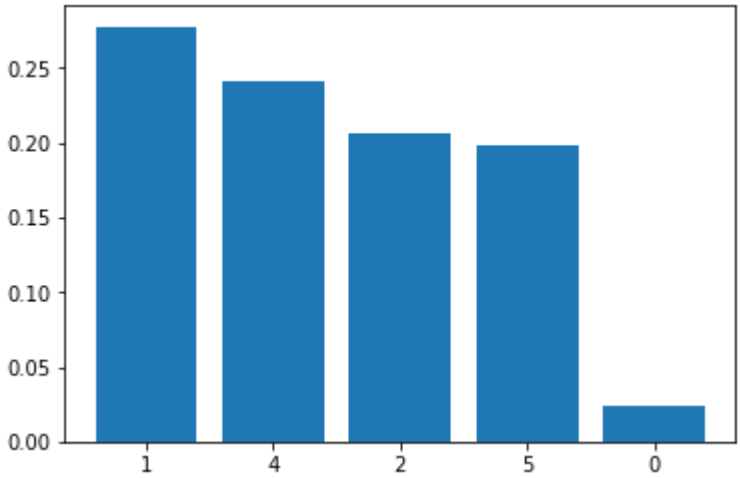

```
The kernel which activates/recognizes the shape: 2
```



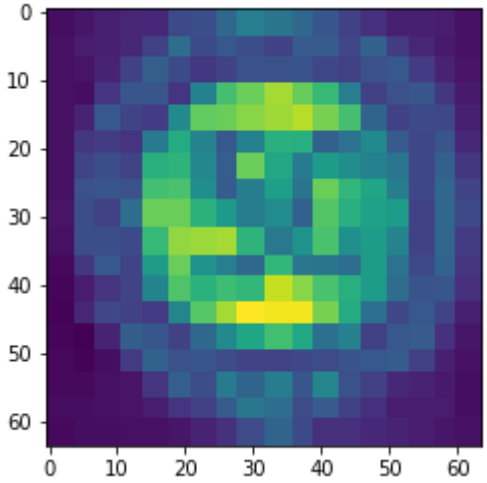
Multiple kernel activations starting from 0



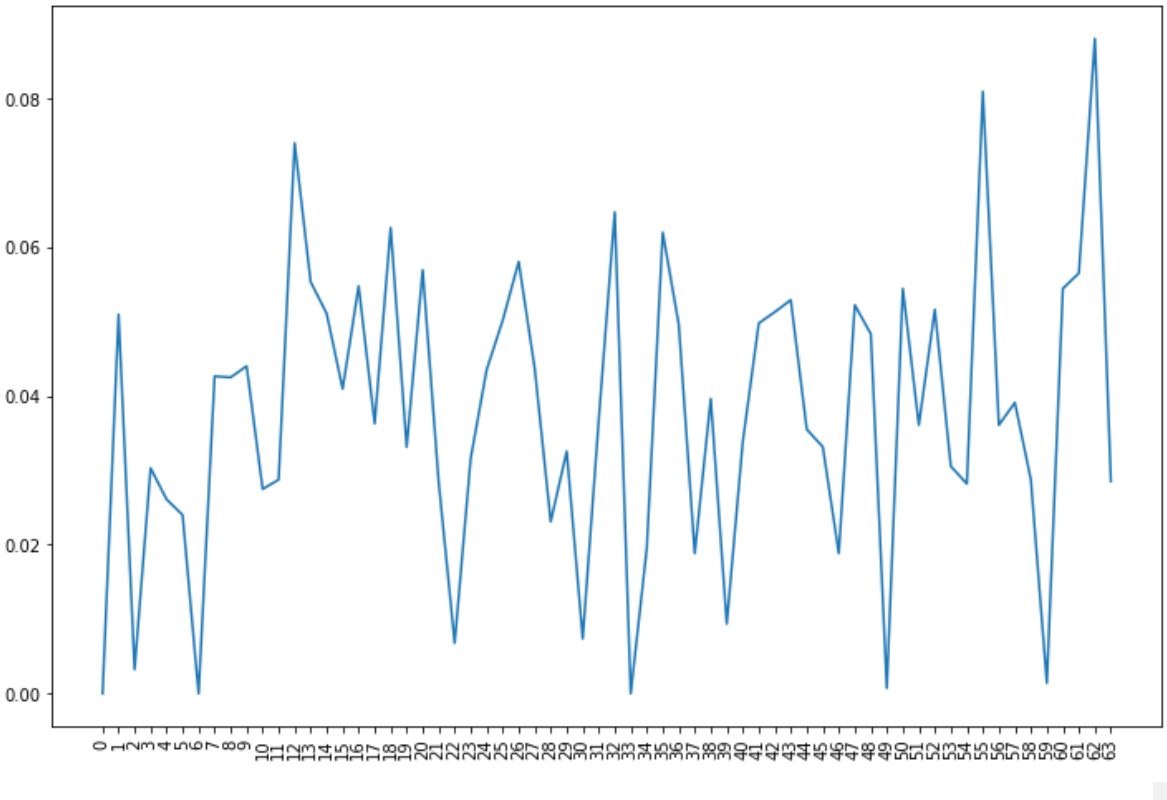
Activation for 2 kernel in 0 layer.

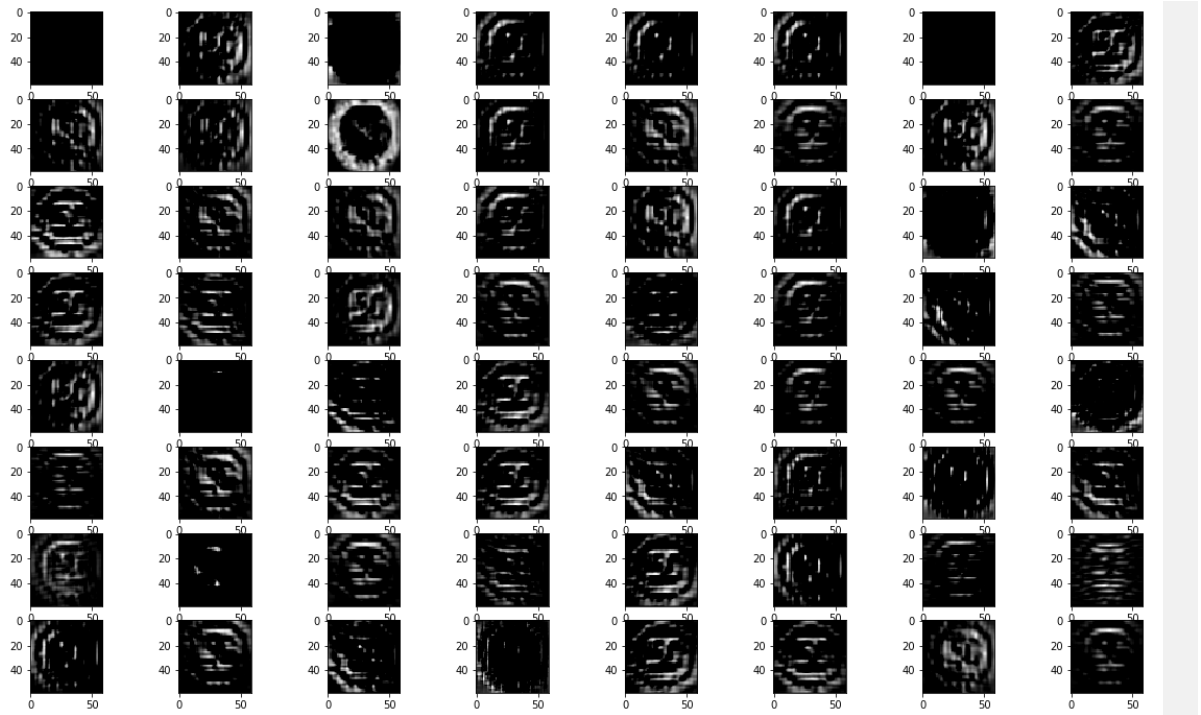



```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 1
Actual training image
```

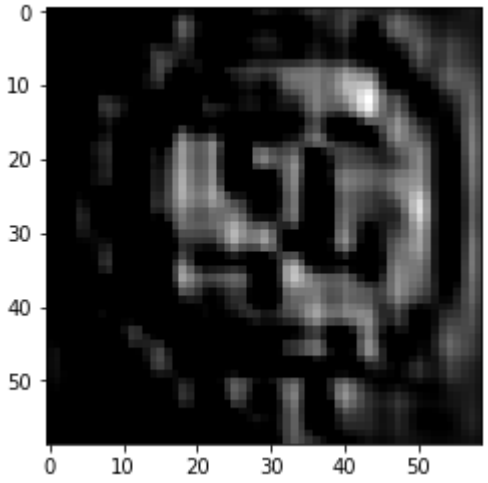



```
The kernel which activates/recognizes the shape: 62
```

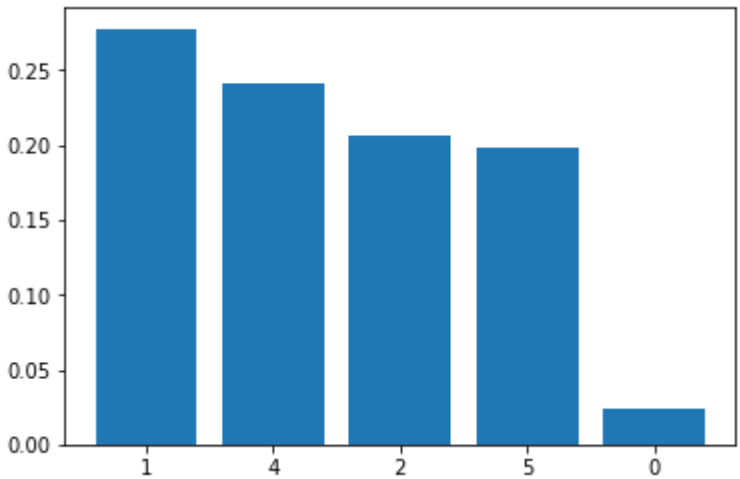




Multiple kernel activations starting from 0



Activation for 62 kernel in 1 layer.



```
Probabilities of top 5 classes.
```

[5.4] Conv(32 -- 4x4) - Conv(64 -- 3x3) - MaxPool(2x2) - Conv(32 -- 4x4) - Dropout(0.75) - Dense(128) - Dropout(0.5)

In [0]:

```
epochs = 15
model = Sequential()
model.add(Conv2D(
    32, kernel_size=(4, 4), activation='relu', input_shape=input_shape
))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (4, 4), activation='relu'))
model.add(Dropout(rate=0.75))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(
    loss=keras.losses.categorical_crossentropy,
    optimizer=keras.optimizers.Adadelta(), metrics=['accuracy']
)

model.summary()

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 61, 61, 32)        544
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 59, 59, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 29, 29, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 26, 26, 32)        32800
_________________________________________________________________
dropout_9 (Dropout)          (None, 26, 26, 32)        0
_________________________________________________________________
flatten_5 (Flatten)          (None, 21632)             0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               2769024
_________________________________________________________________
dropout_10 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 43)                5547
=================================================================
Total params: 2,826,411
Trainable params: 2,826,411
Non-trainable params: 0
_________________________________________________________________
Train on 27446 samples, validate on 11763 samples
Epoch 1/15
27446/27446 [==============================] - 10s 355us/step - loss: 2.0574
- acc: 0.4623 - val_loss: 0.6537 - val_acc: 0.8741
Epoch 2/15
27446/27446 [==============================] - 9s 323us/step - loss: 0.5429
- acc: 0.8494 - val_loss: 0.2080 - val_acc: 0.9562
Epoch 3/15
```

```
27446/27446 [==============================] - 9s 326us/step - loss: 0.3495
- acc: 0.8990 - val_loss: 0.1442 - val_acc: 0.9640
Epoch 4/15
27446/27446 [==============================] - 9s 329us/step - loss: 0.2650
- acc: 0.9244 - val_loss: 0.1023 - val_acc: 0.9771
Epoch 5/15
27446/27446 [==============================] - 9s 331us/step - loss: 0.2069
- acc: 0.9374 - val_loss: 0.0846 - val_acc: 0.9795
Epoch 6/15
27446/27446 [==============================] - 9s 333us/step - loss: 0.1751
- acc: 0.9491 - val_loss: 0.0768 - val_acc: 0.9802
Epoch 7/15
27446/27446 [==============================] - 9s 332us/step - loss: 0.1563
- acc: 0.9539 - val_loss: 0.0675 - val_acc: 0.9840
Epoch 8/15
27446/27446 [==============================] - 9s 333us/step - loss: 0.1361
- acc: 0.9607 - val_loss: 0.0700 - val_acc: 0.9816
Epoch 9/15
27446/27446 [==============================] - 9s 331us/step - loss: 0.1253
- acc: 0.9628 - val_loss: 0.0703 - val_acc: 0.9833
Epoch 10/15
27446/27446 [==============================] - 9s 328us/step - loss: 0.1150
- acc: 0.9657 - val_loss: 0.0574 - val_acc: 0.9869
Epoch 11/15
27446/27446 [==============================] - 9s 327us/step - loss: 0.1054
- acc: 0.9684 - val_loss: 0.0528 - val_acc: 0.9880
Epoch 12/15
27446/27446 [==============================] - 9s 327us/step - loss: 0.0983
- acc: 0.9712 - val_loss: 0.0543 - val_acc: 0.9878
Epoch 13/15
27446/27446 [==============================] - 9s 328us/step - loss: 0.0888
- acc: 0.9727 - val_loss: 0.0485 - val_acc: 0.9876
Epoch 14/15
27446/27446 [==============================] - 9s 328us/step - loss: 0.0878
- acc: 0.9727 - val_loss: 0.0469 - val_acc: 0.9888
Epoch 15/15
27446/27446 [==============================] - 9s 328us/step - loss: 0.0776
- acc: 0.9756 - val_loss: 0.0470 - val_acc: 0.9893
```

In [0]:

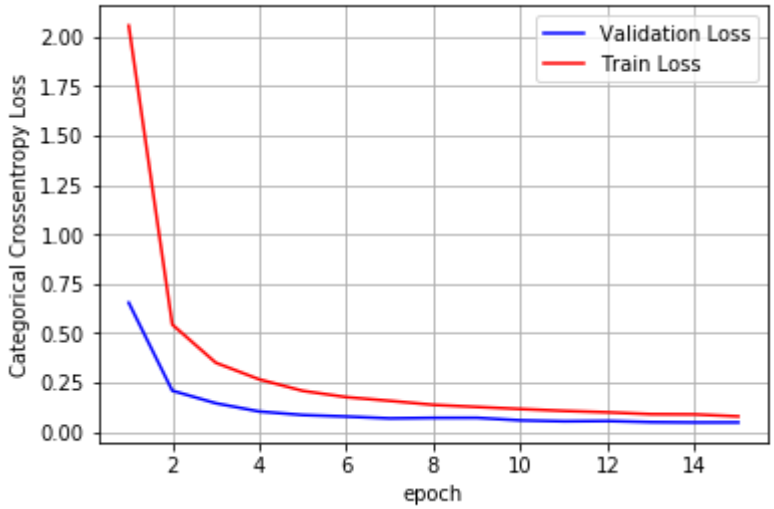
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(fig, x, vy, ty, ax)
```

```
Test score: 0.04703236871649887
Test accuracy: 0.9892884468247896
```



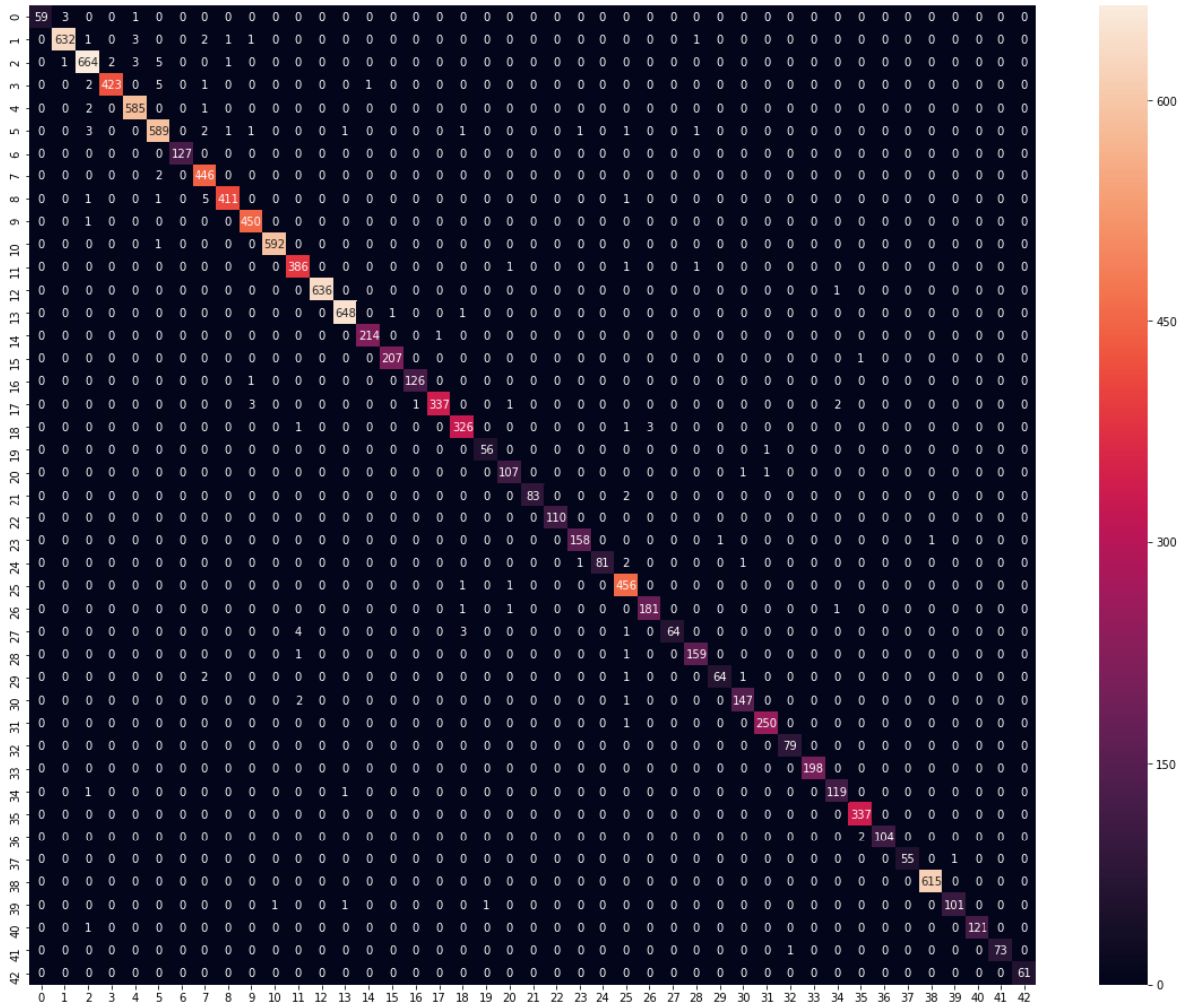
In [0]:

```
plt.figure(figsize=(20,16))
y_prediction = model.predict(x_test)
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_prediction, axis = 1)
# Convert validation observations to one hot vectors
y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
print("-----------------------------------------------------------")
print("Micro F1 score", f1_score(y_true, y_pred_classes, average='micro'))
print("-----------------------------------------------------------")
confusion_mtx = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(confusion_mtx, annot=True, fmt="d")
```

```
-----------------------------------------------------------
Micro F1 score 0.9892884468247896
-----------------------------------------------------------
```

Out[92]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b00f22588>
```



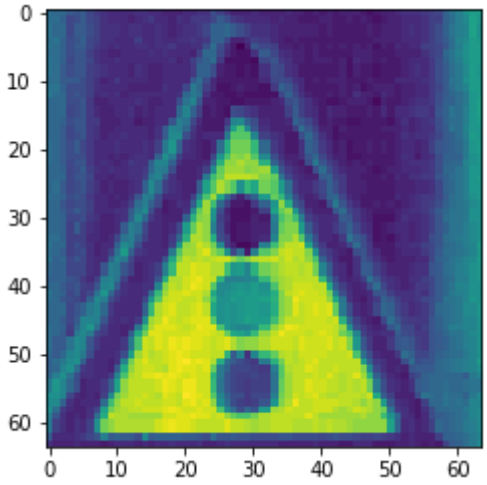
In [0]:

```
layer_outputs = [layer.output for layer in model.layers]
activation_model = Model(inputs=model.input, outputs=layer_outputs)
```

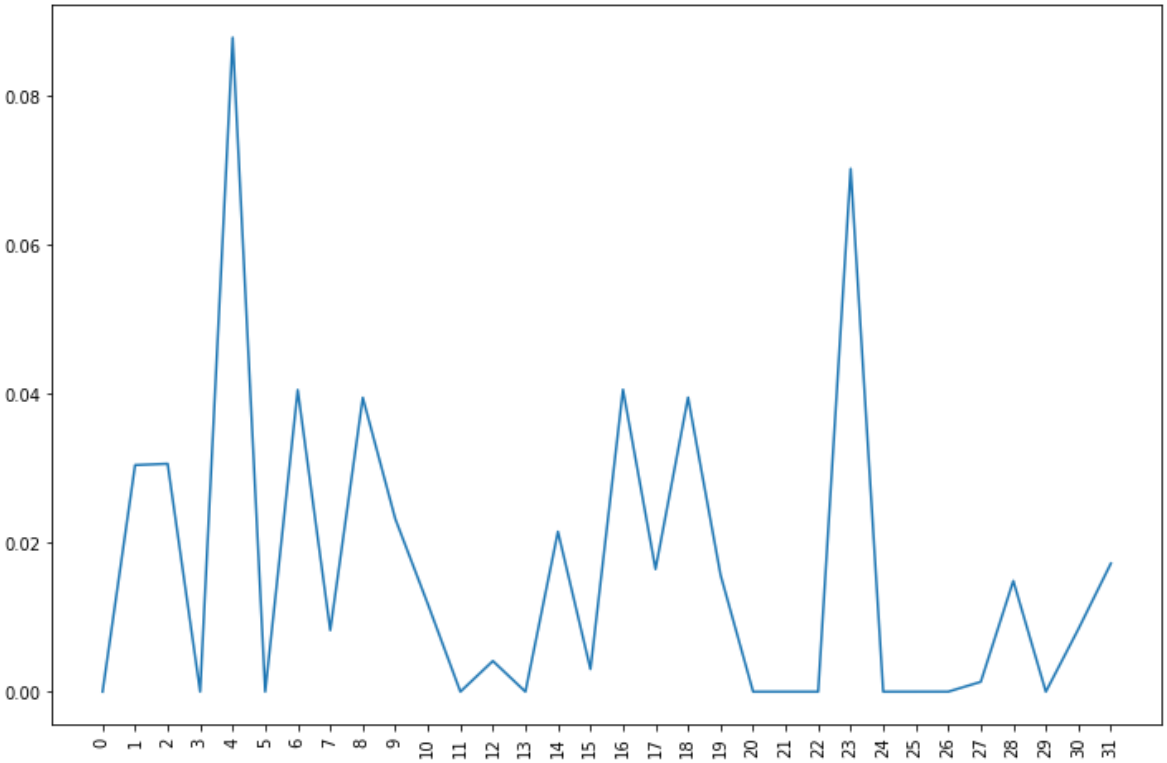
In [0]:

```
idx = 684
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 2, y_true, y_prediction)
```

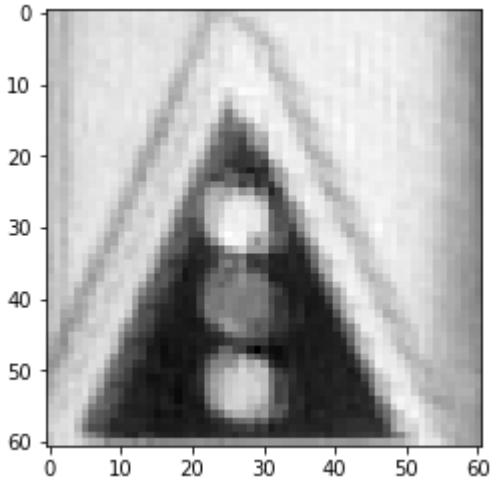
```
Actual class it belongs to: 26
Predicted class 26
Actual training image
```



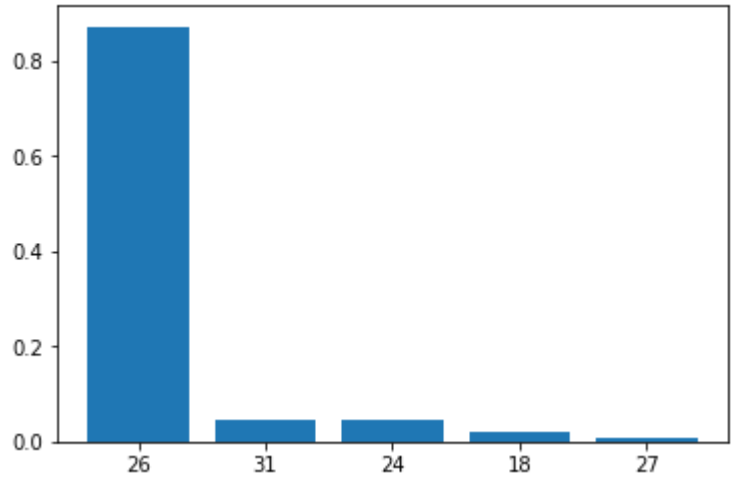

```
The kernel which activates/recognizes the shape: 4
```



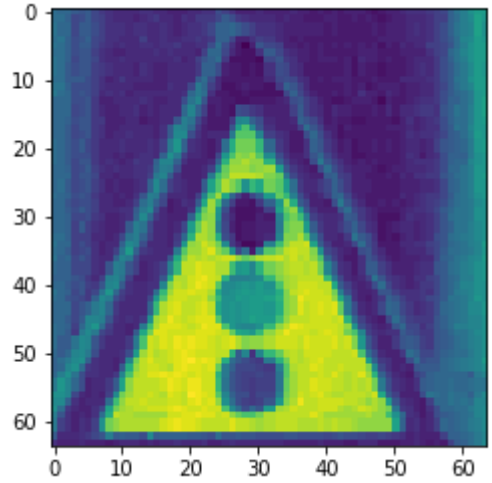
Multiple kernel activations starting from 0



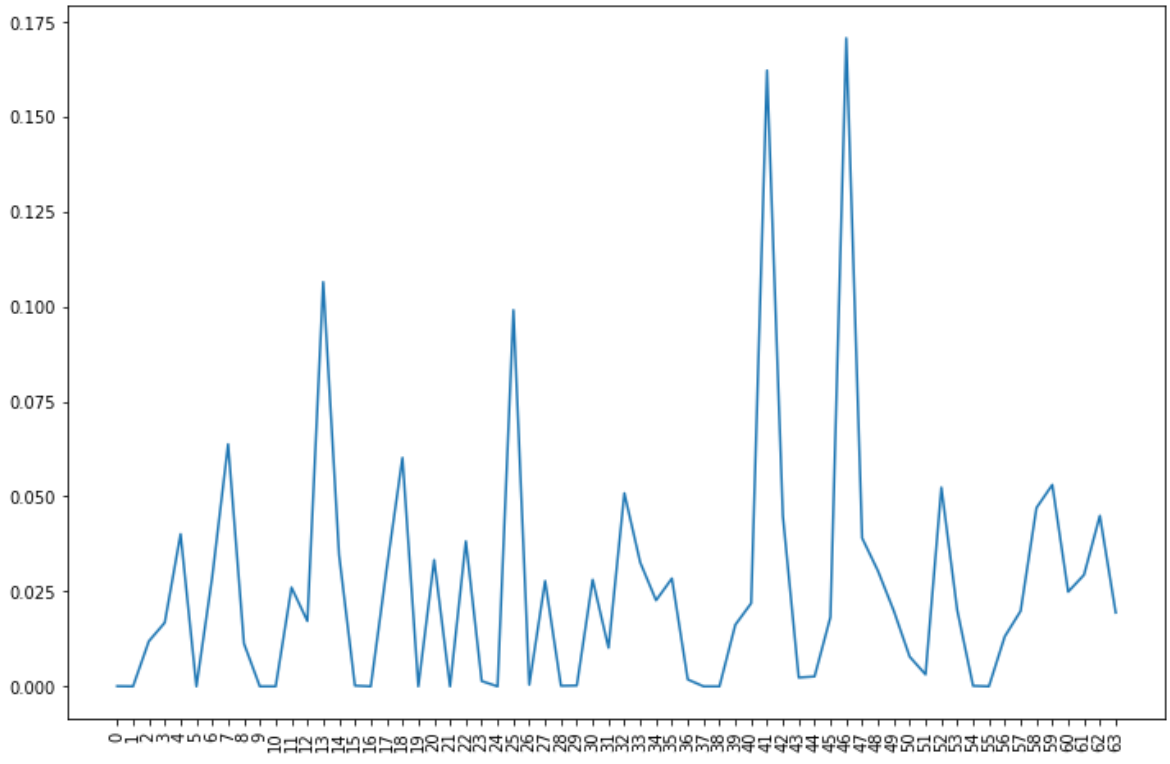
Activation for 4 kernel in 0 layer.

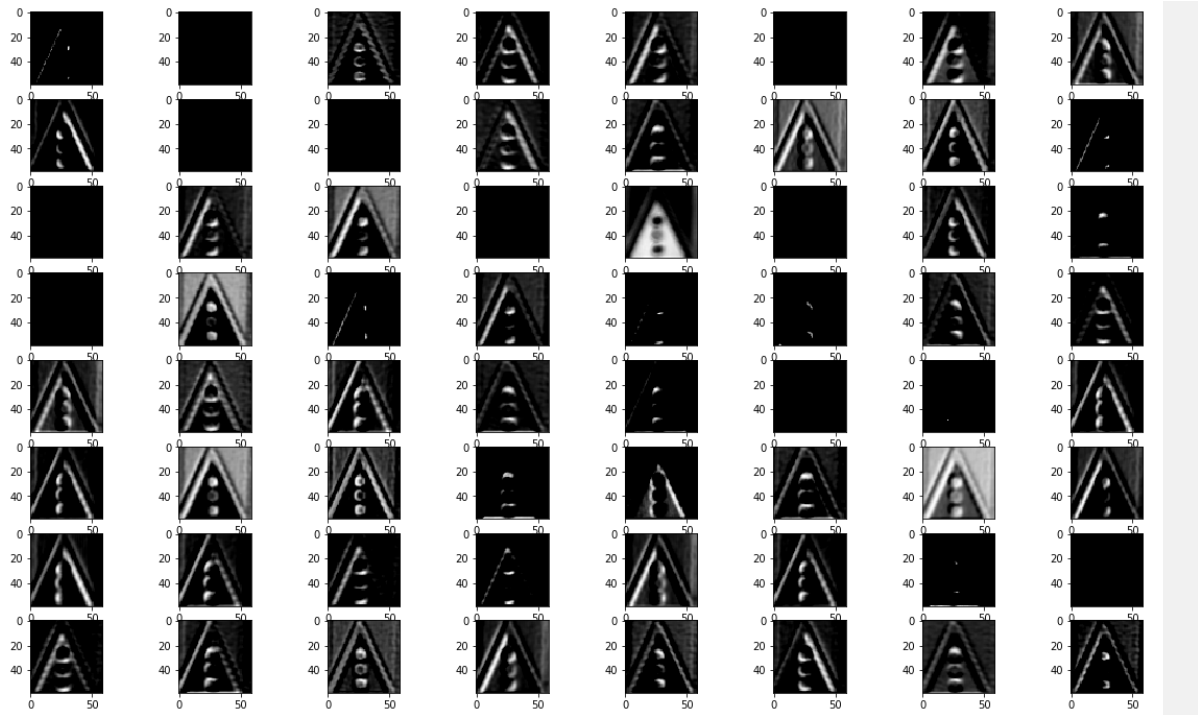



```
Probabilities of top 5 classes.
Actual class it belongs to: 26
Predicted class 26
Actual training image
```

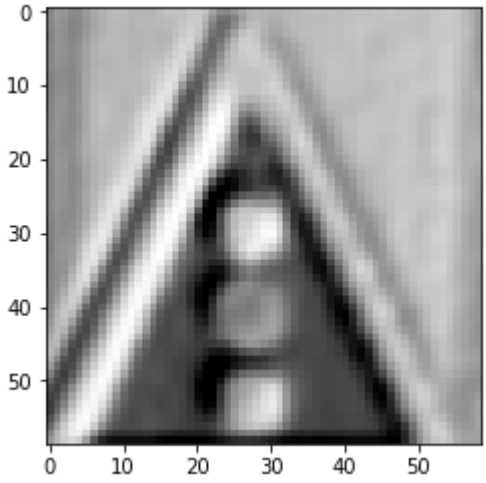



```
The kernel which activates/recognizes the shape: 46
```

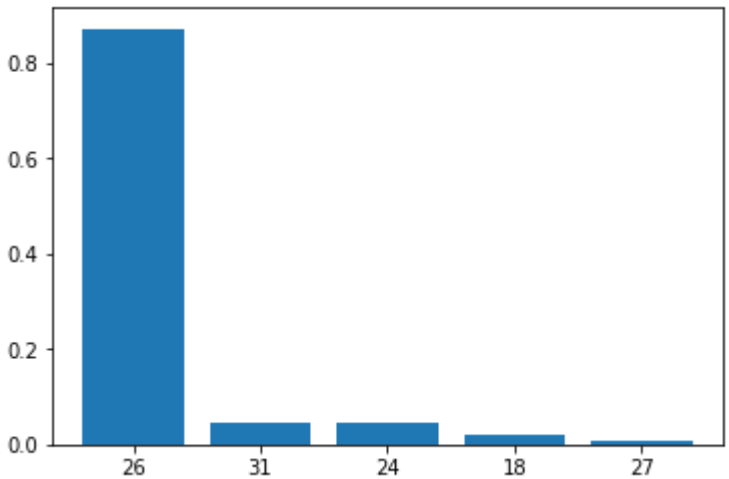




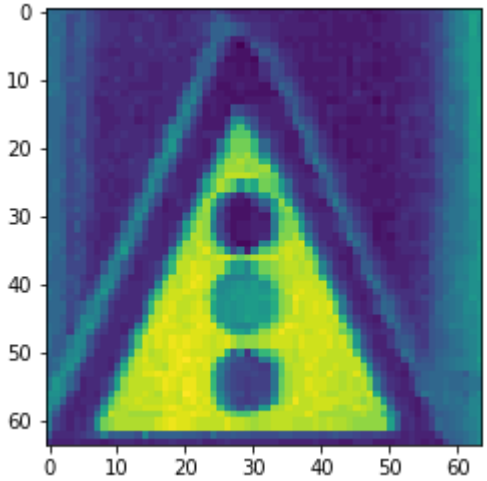
Multiple kernel activations starting from 0



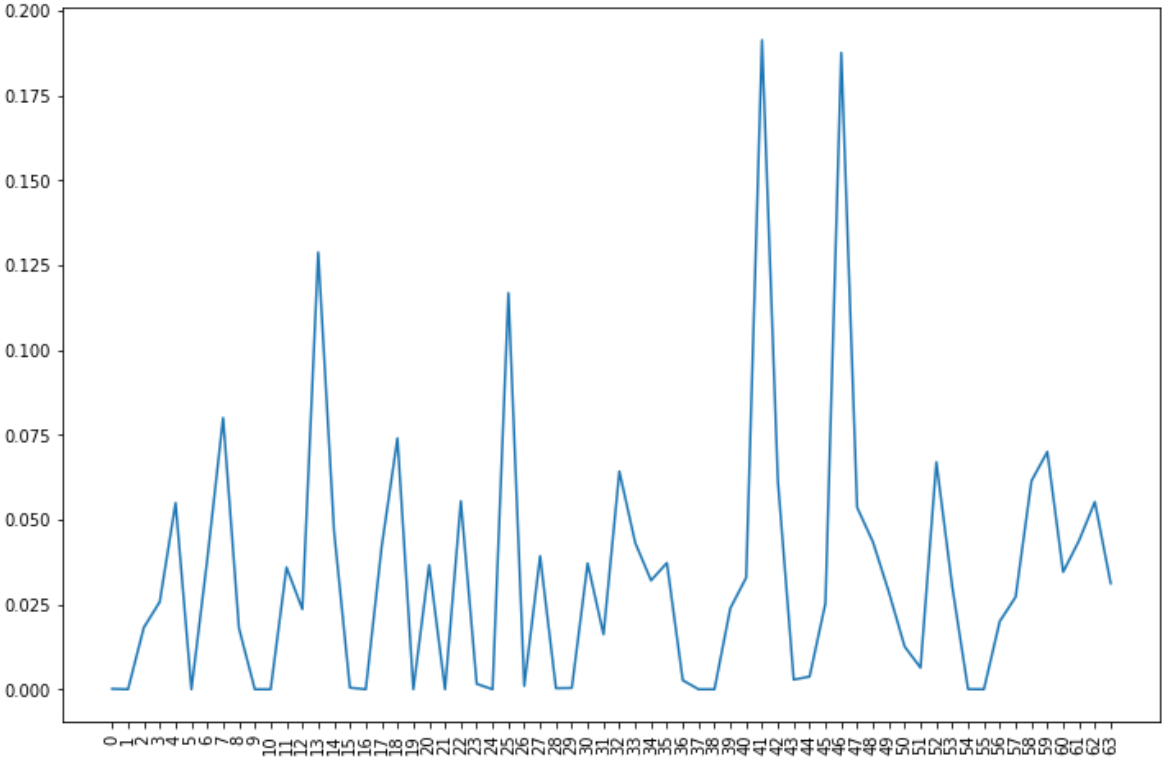
Activation for 46 kernel in 1 layer.



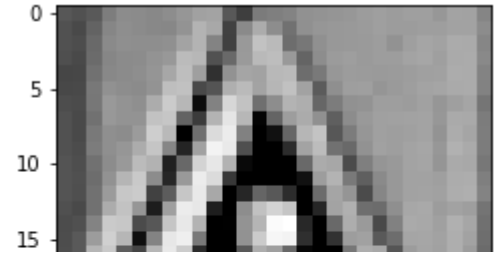
```
Probabilities of top 5 classes.
Actual class it belongs to: 26
Predicted class 26
Actual training image
```



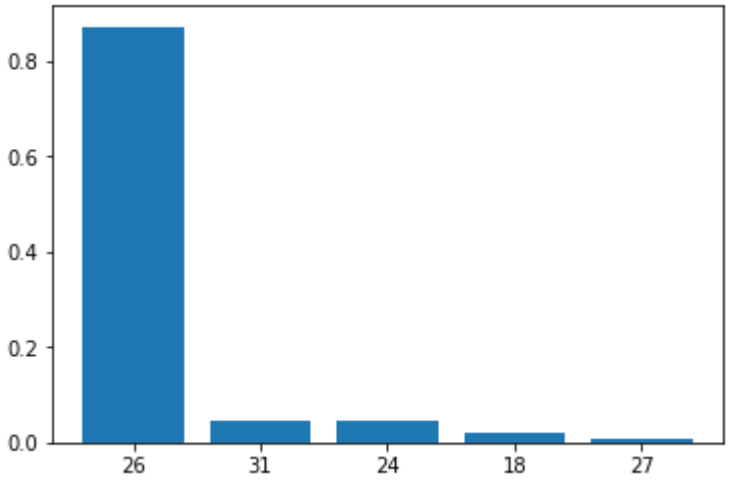

```
The kernel which activates/recognizes the shape: 41
```



Multiple kernel activations starting from 0



Activation for 41 kernel in 2 layer.

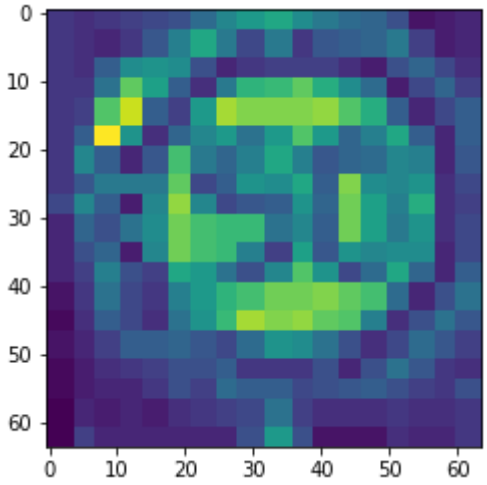


Probabilities of top 5 classes.

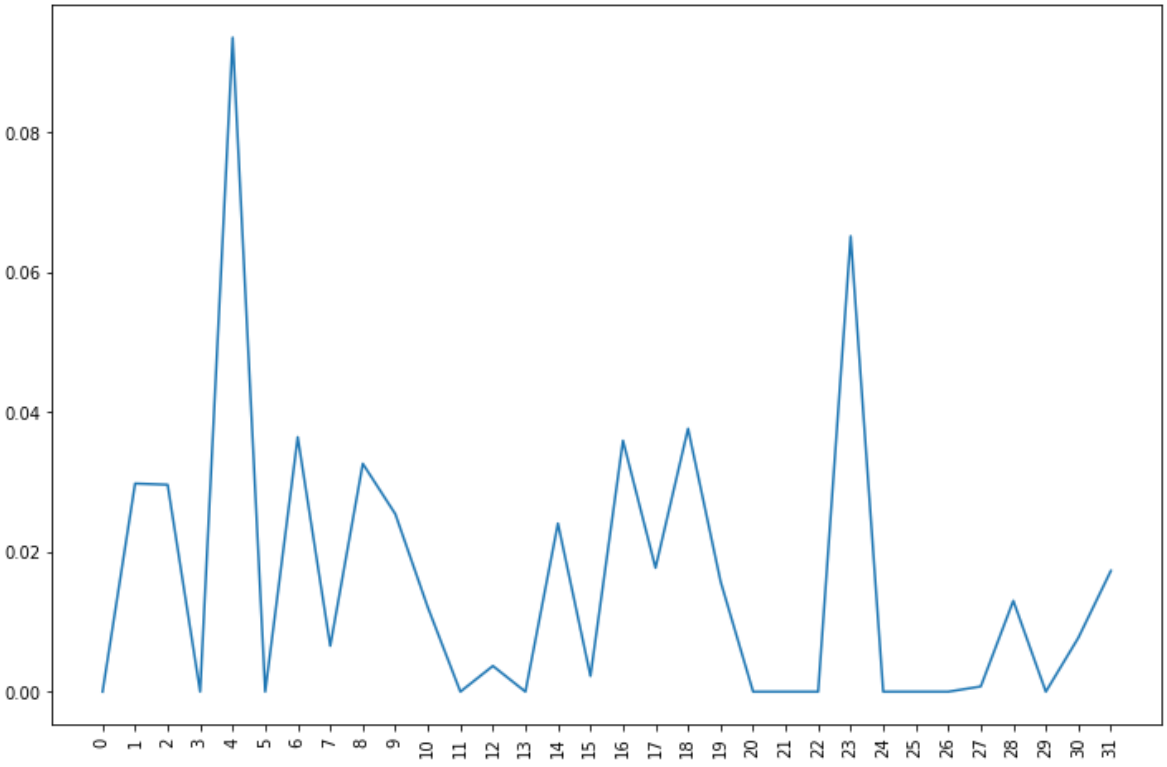
In [0]:

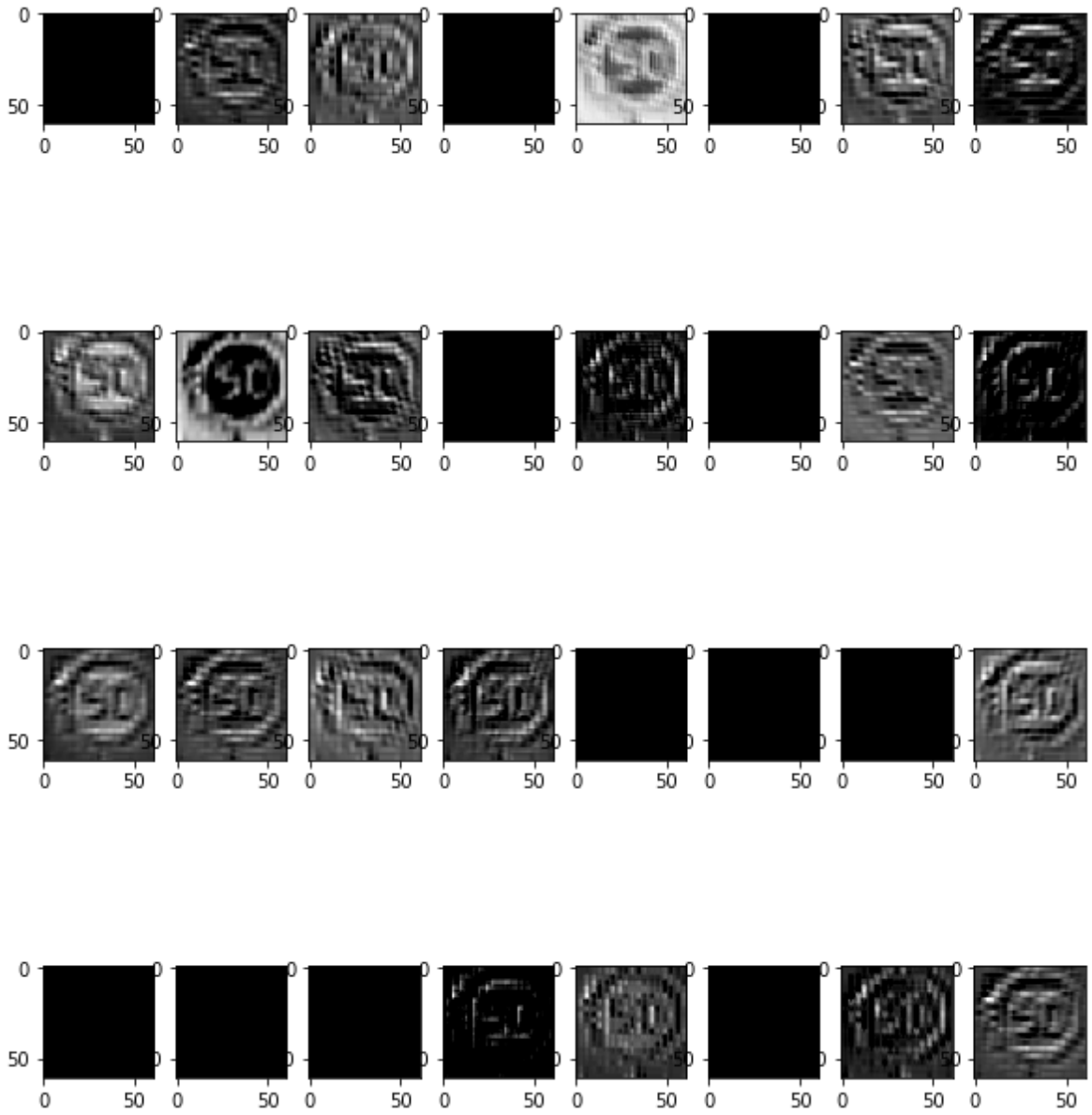
```
idx = 11078
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 2, y_true, y_prediction)
```

```
Actual class it belongs to: 2
Predicted class 4
Actual training image
```

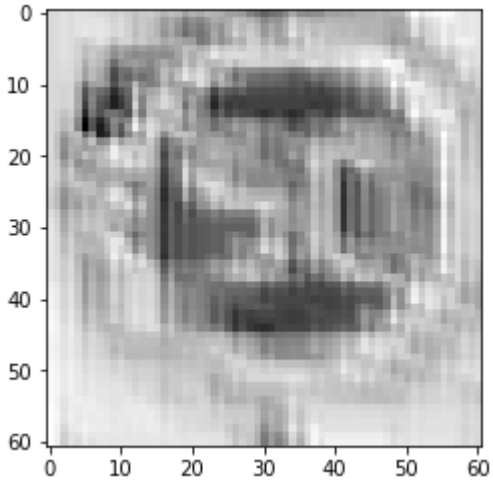



```
The kernel which activates/recognizes the shape: 4
```

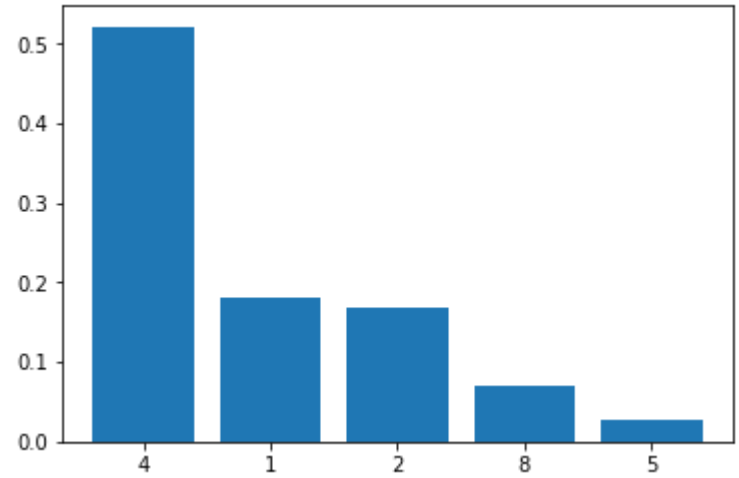




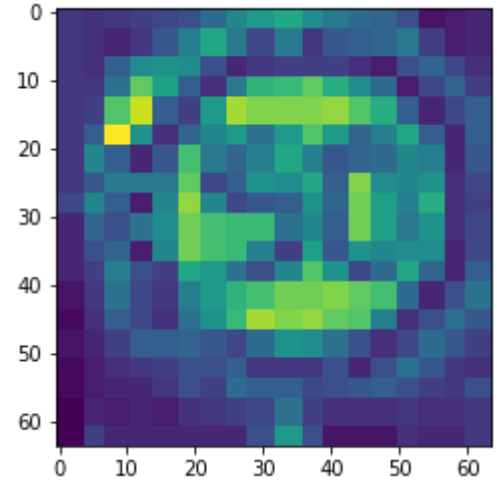
Multiple kernel activations starting from 0



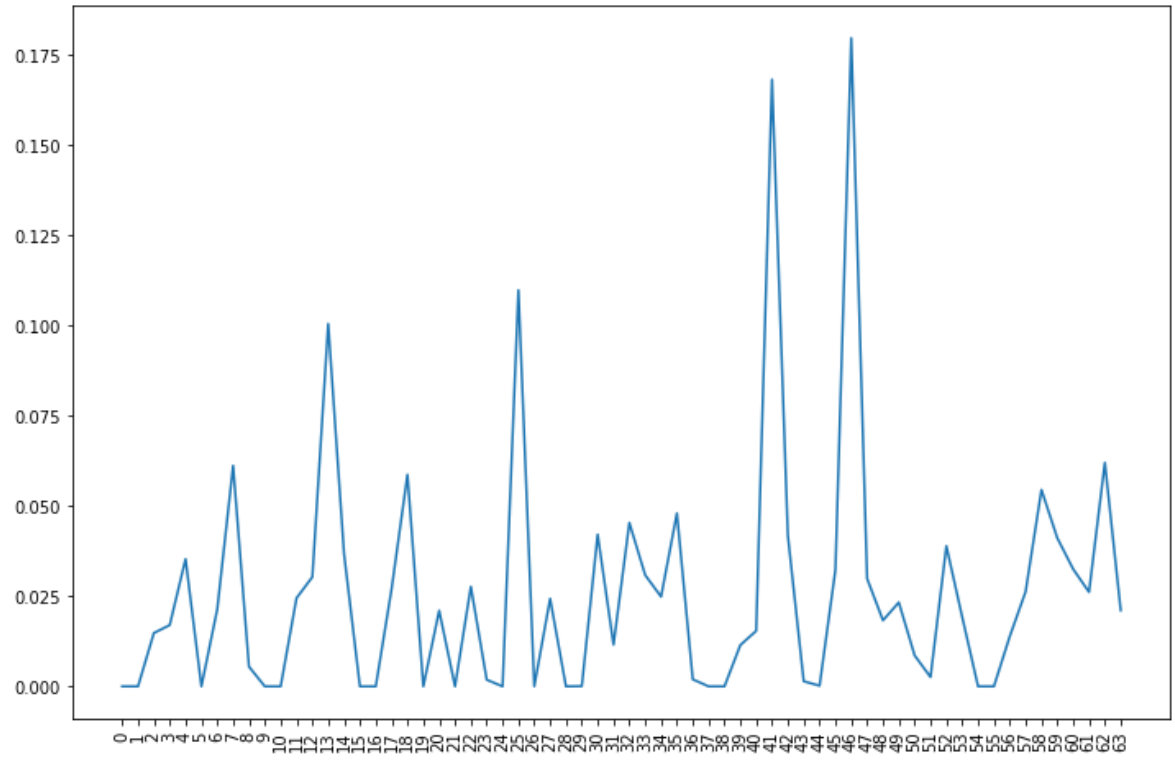
Activation for 4 kernel in 0 layer.

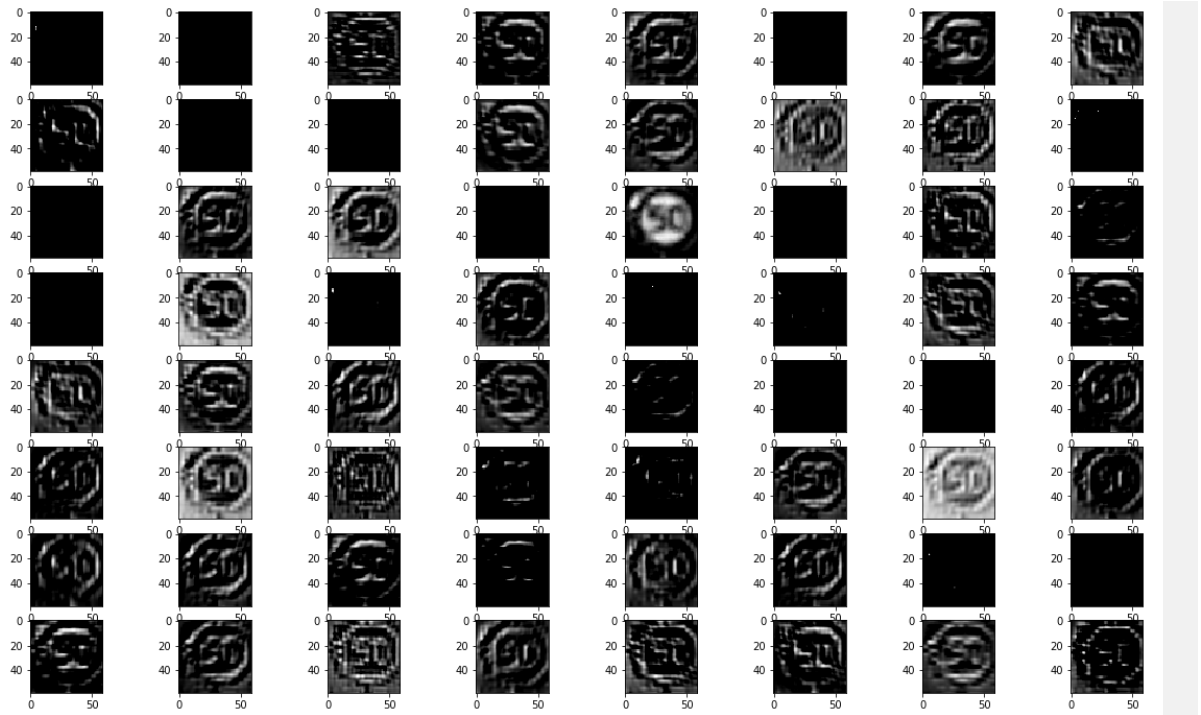



```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 4
Actual training image
```

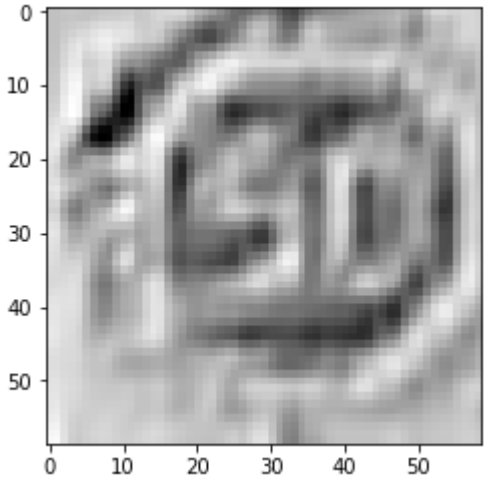



```
The kernel which activates/recognizes the shape: 46
```

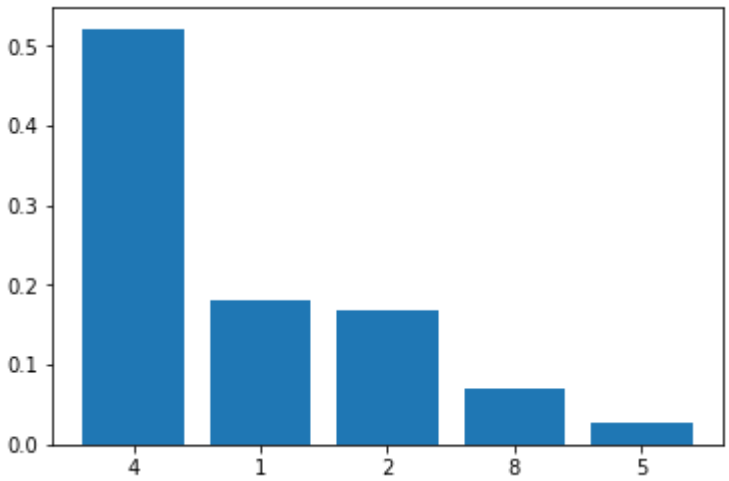




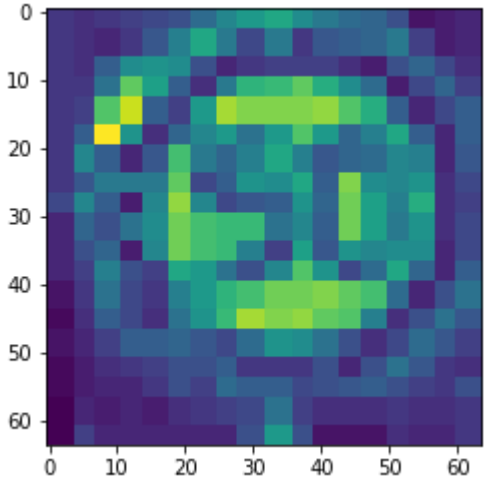
Multiple kernel activations starting from 0



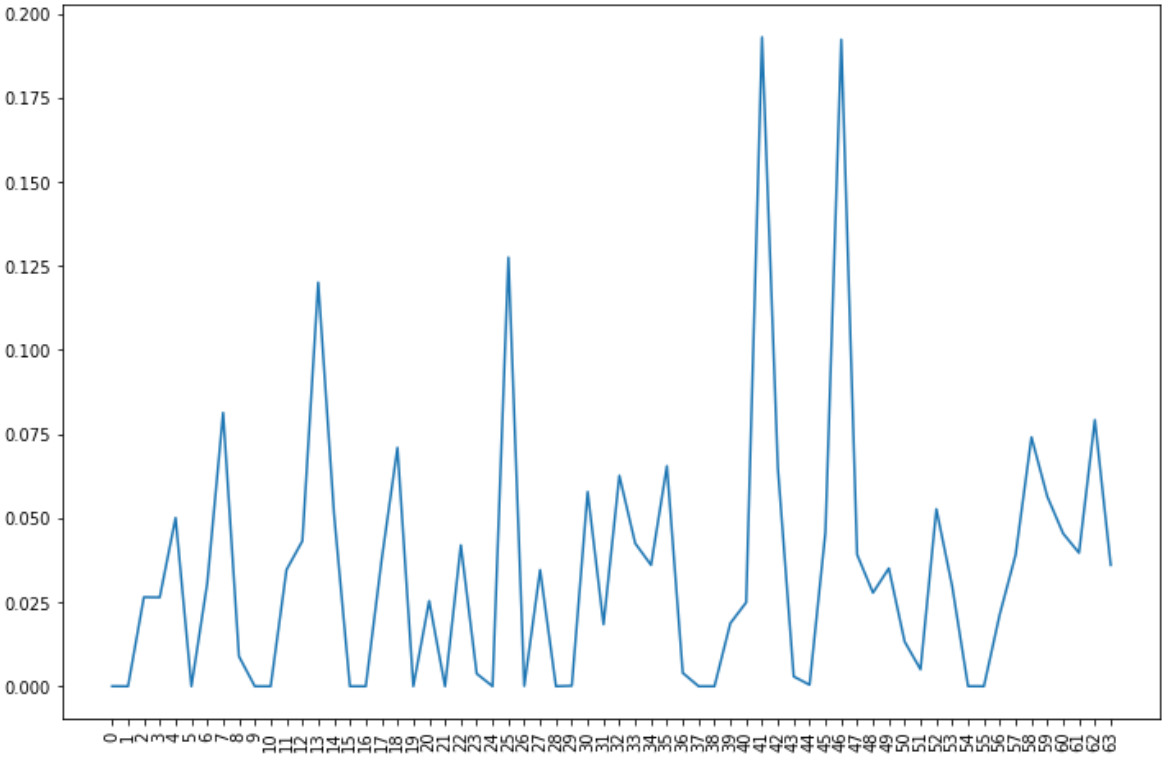
Activation for 46 kernel in 1 layer.

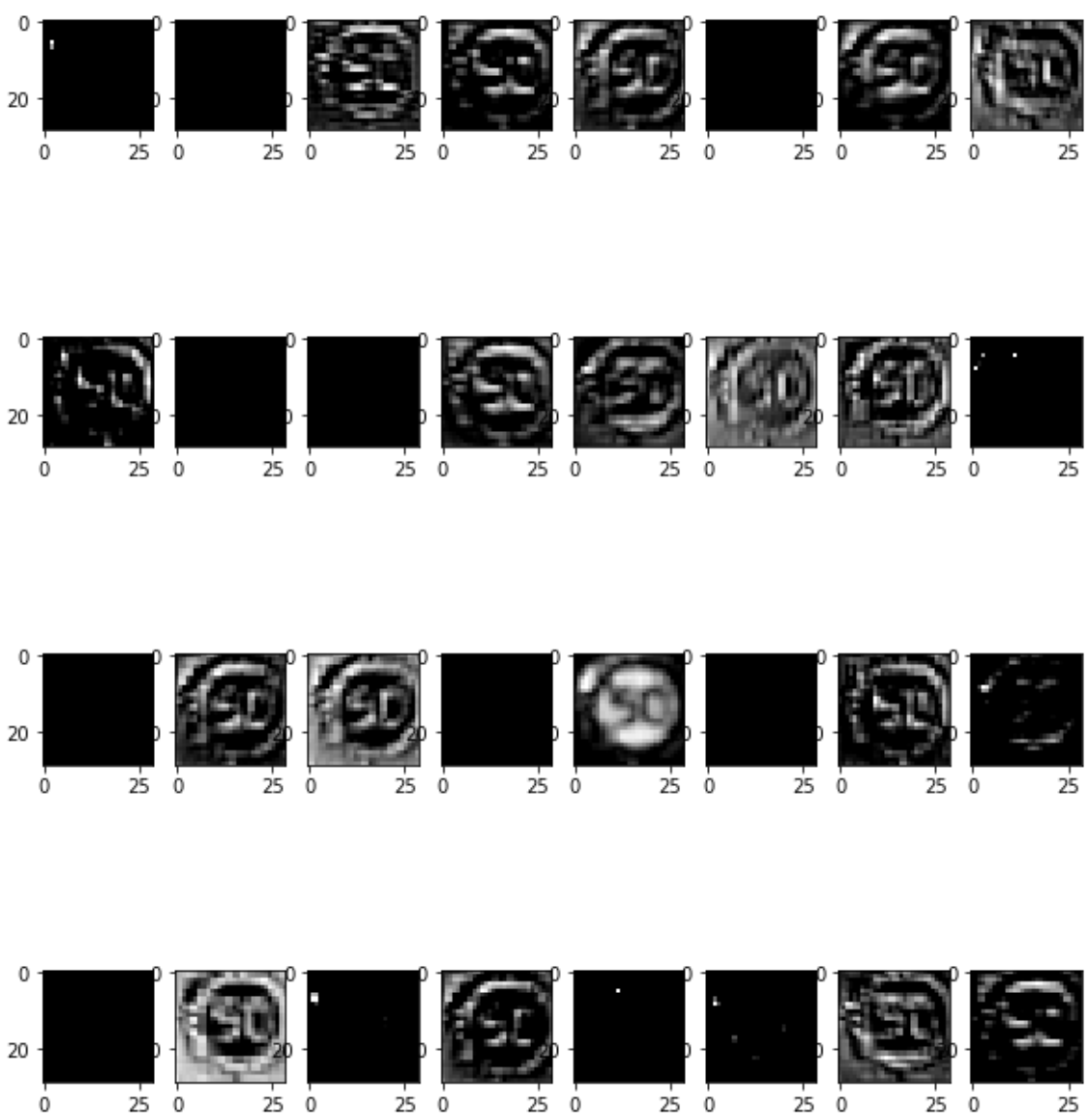


```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 4
Actual training image
```

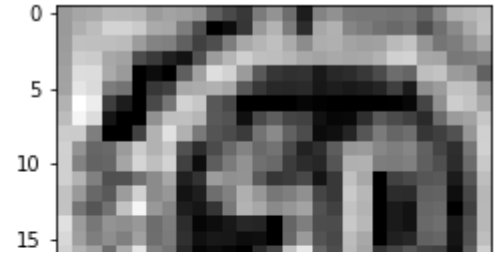



```
The kernel which activates/recognizes the shape: 41
```

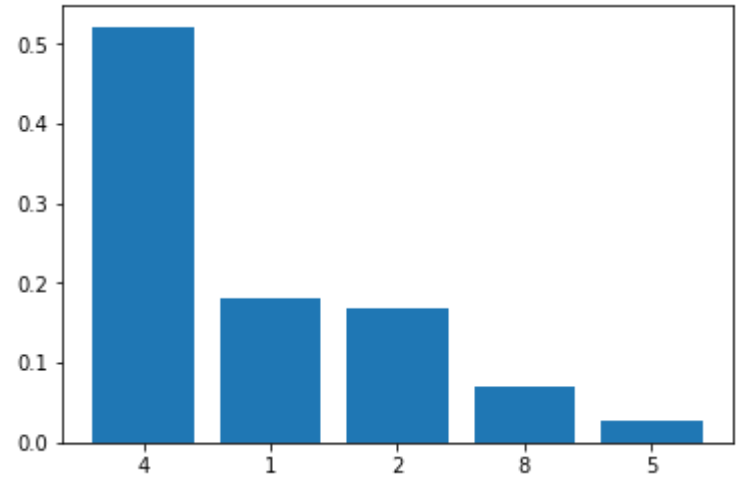




Multiple kernel activations starting from 0



Activation for 41 kernel in 2 layer.

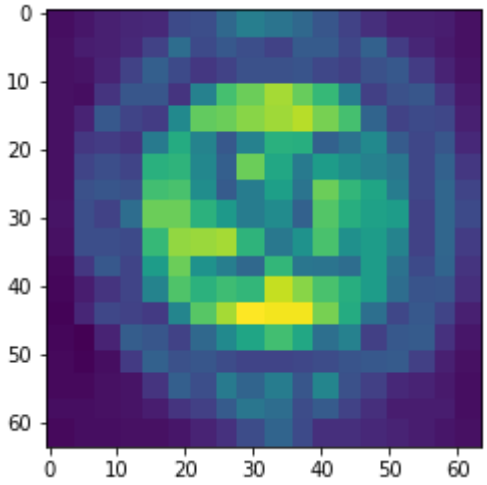


Probabilities of top 5 classes.

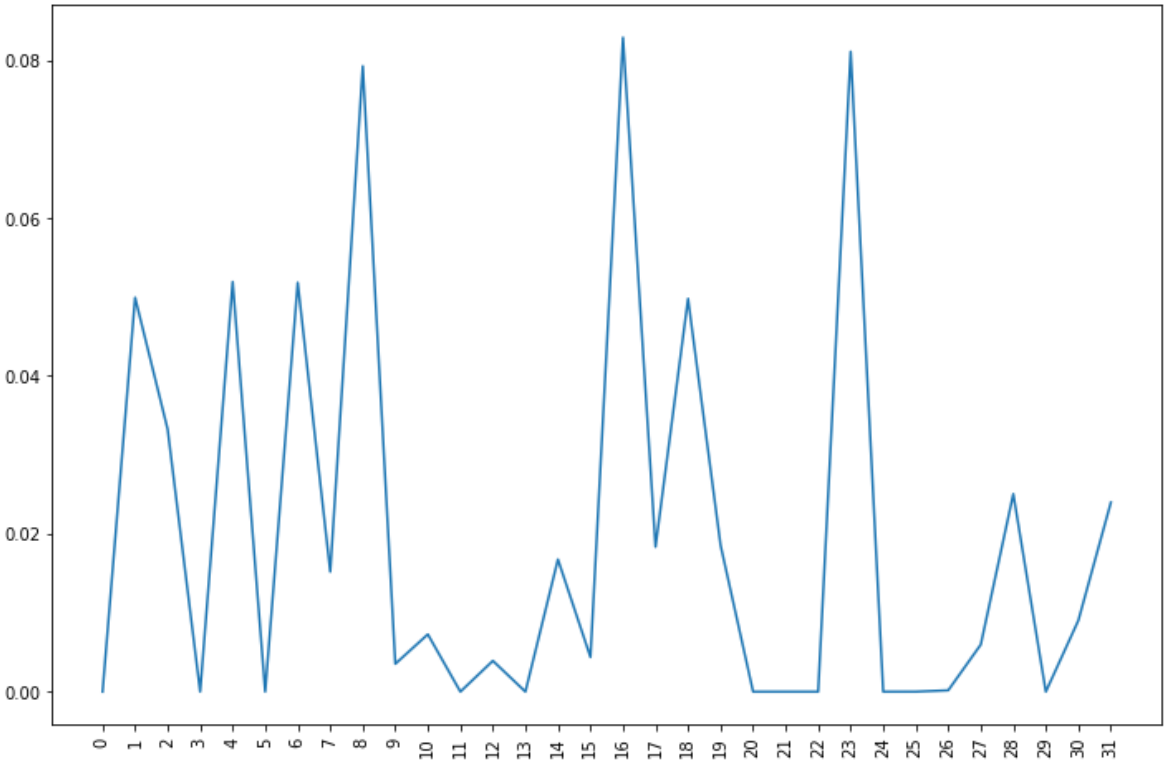
In [0]:

```
idx = 11527
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 2, y_true, y_prediction)
```

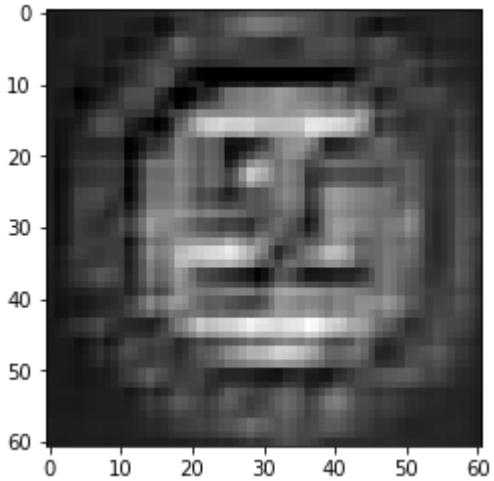
```
Actual class it belongs to: 2
Predicted class 2
Actual training image
```



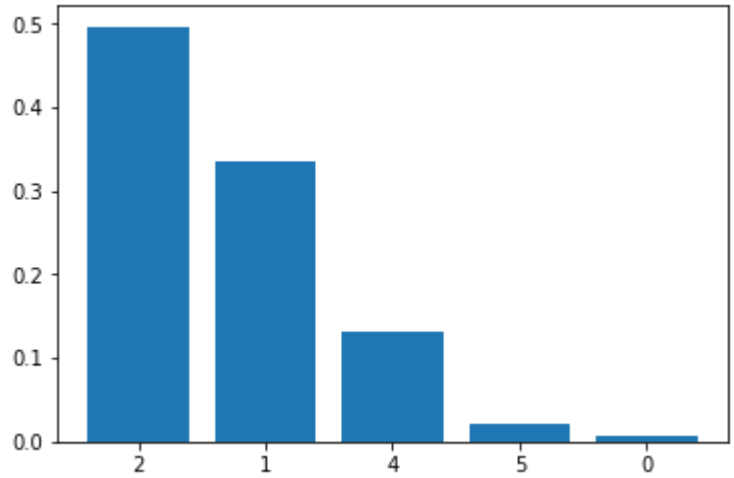

```
The kernel which activates/recognizes the shape: 16
```



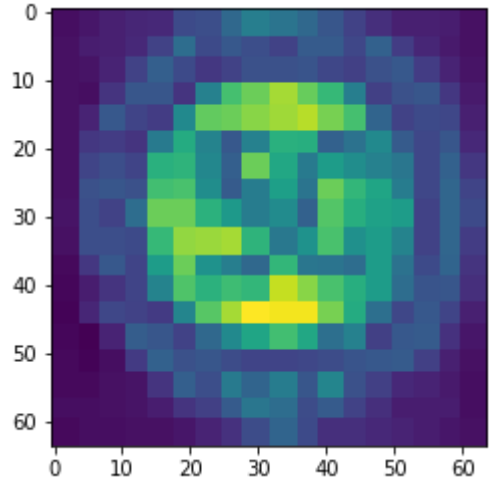
Multiple kernel activations starting from 0



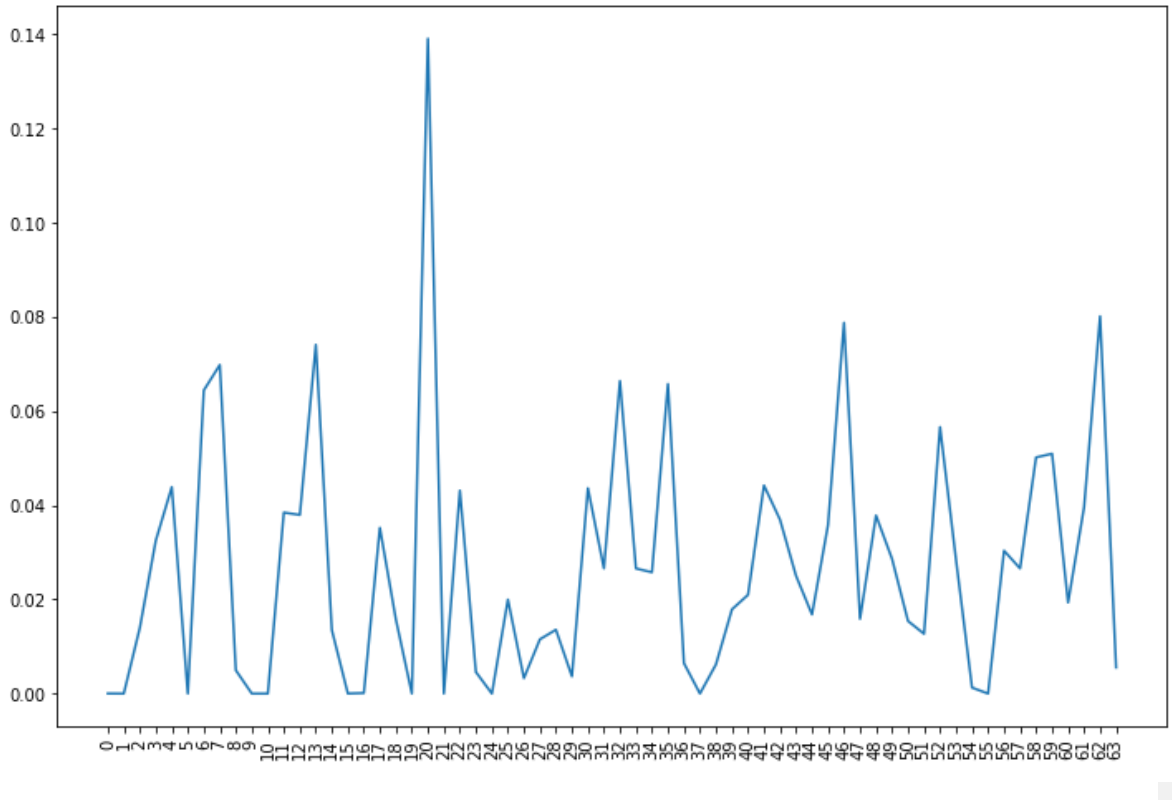
Activation for 16 kernel in 0 layer.

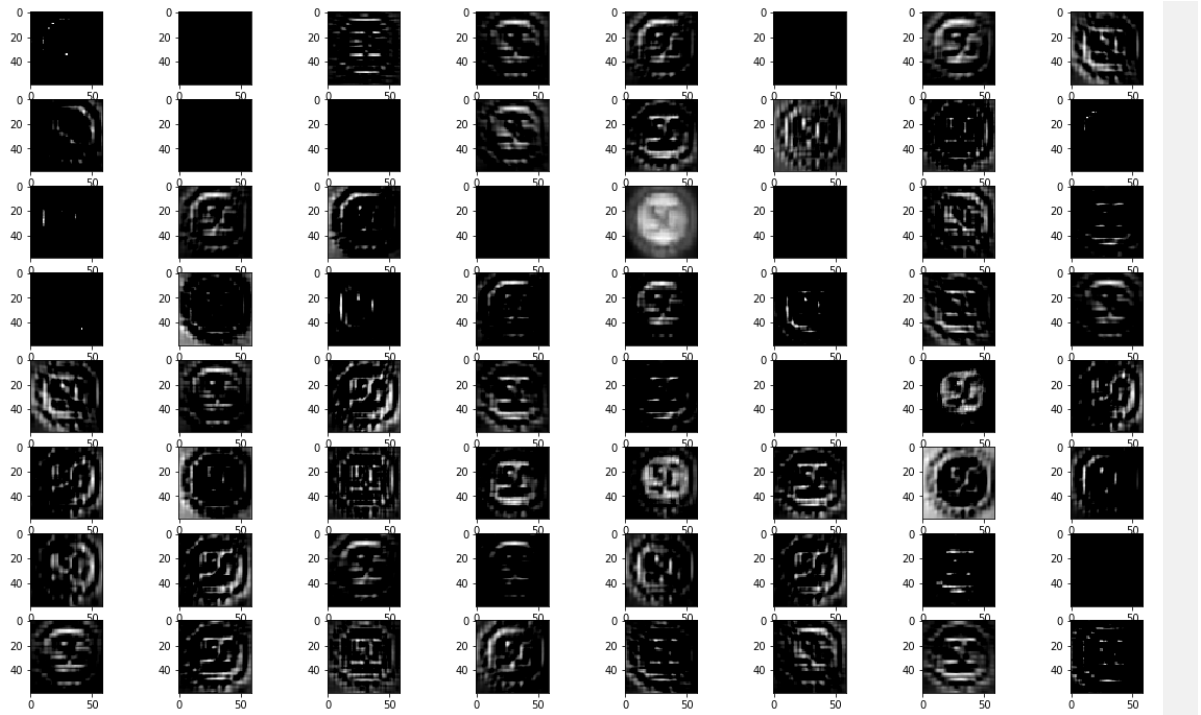



```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 2
Actual training image
```

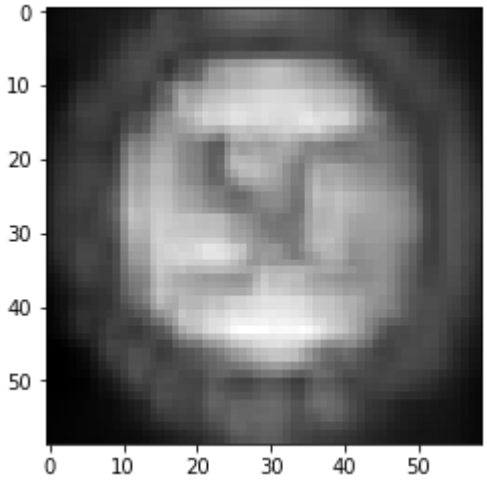



```
The kernel which activates/recognizes the shape: 20
```

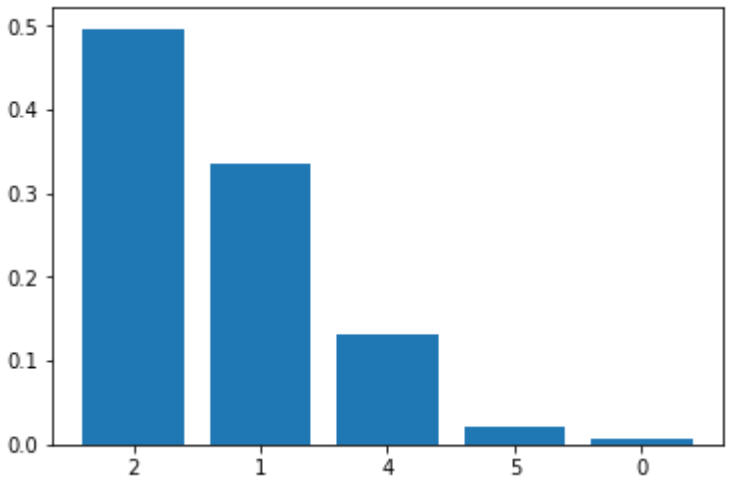




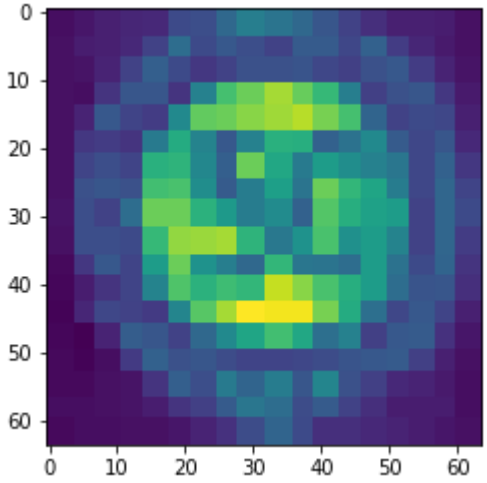
Multiple kernel activations starting from 0



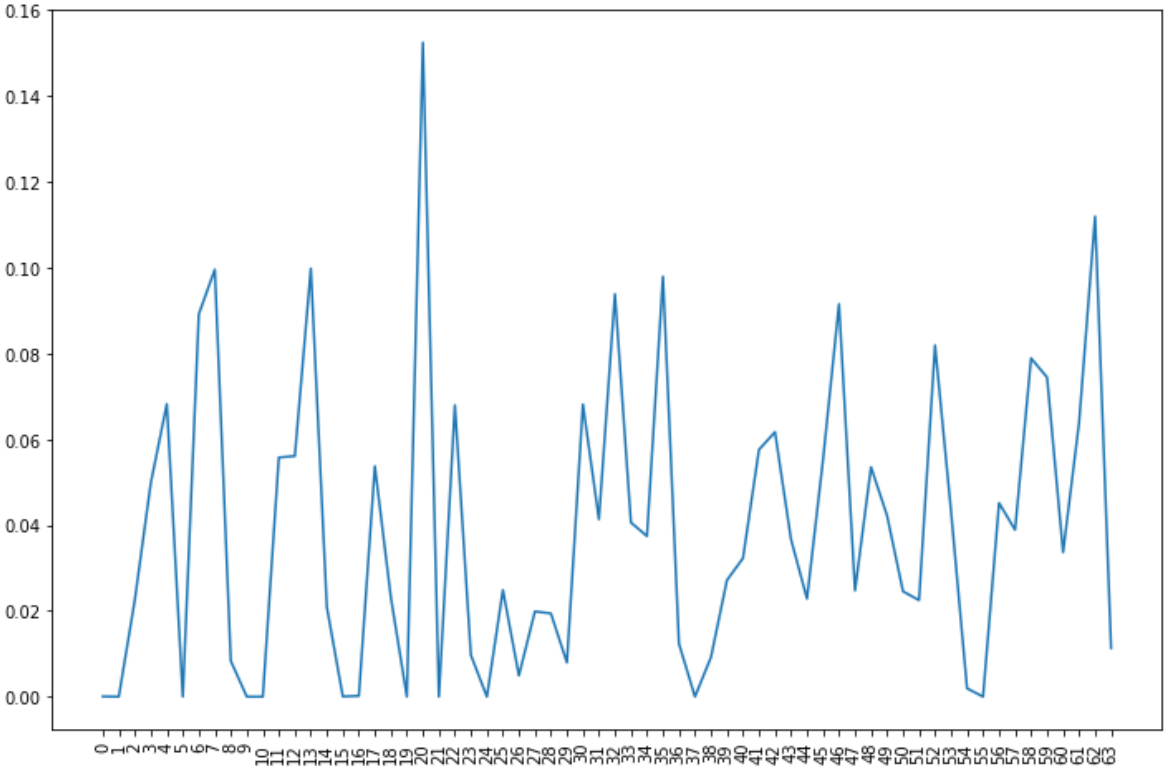
Activation for 20 kernel in 1 layer.

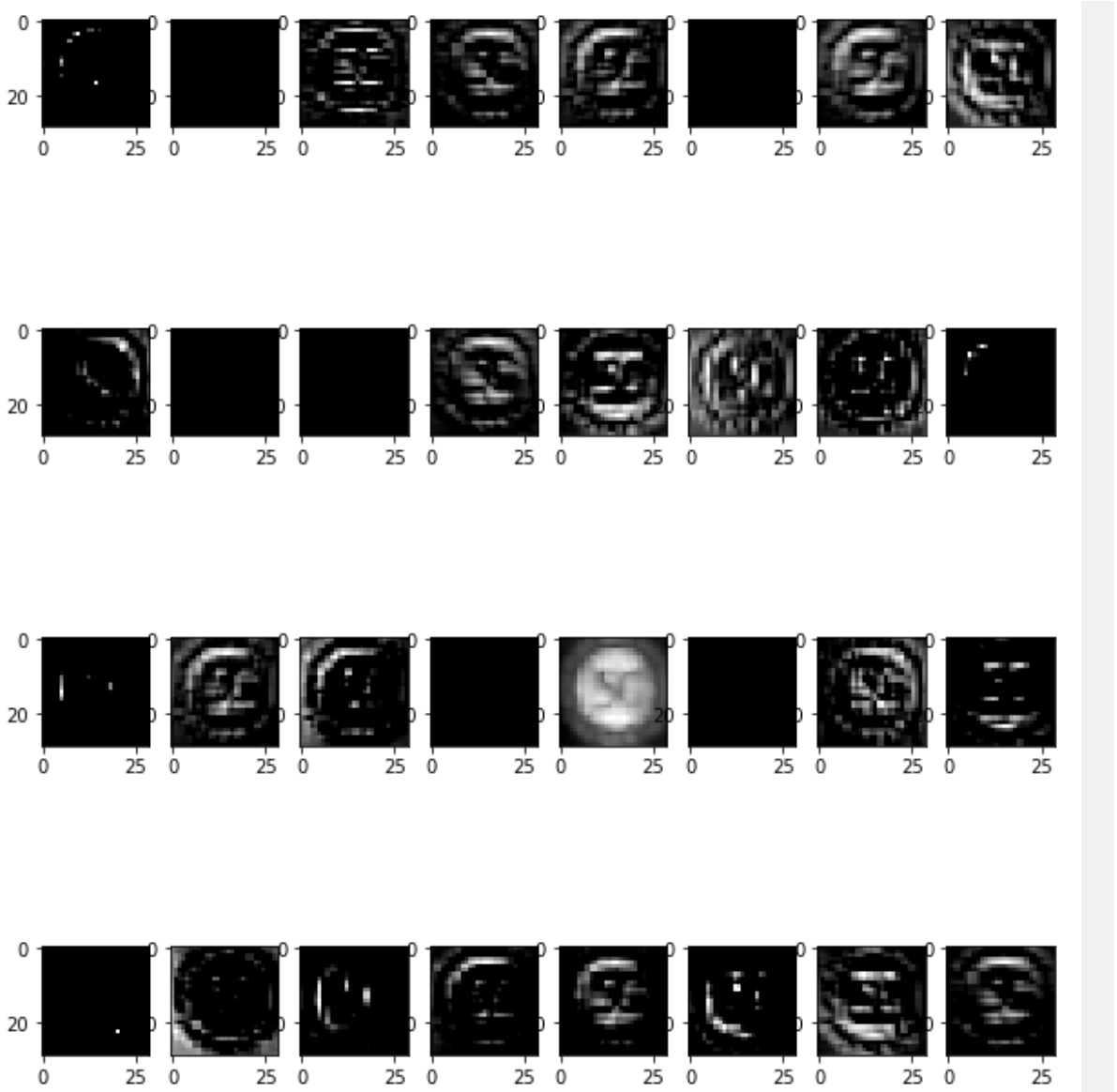


```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 2
Actual training image
```



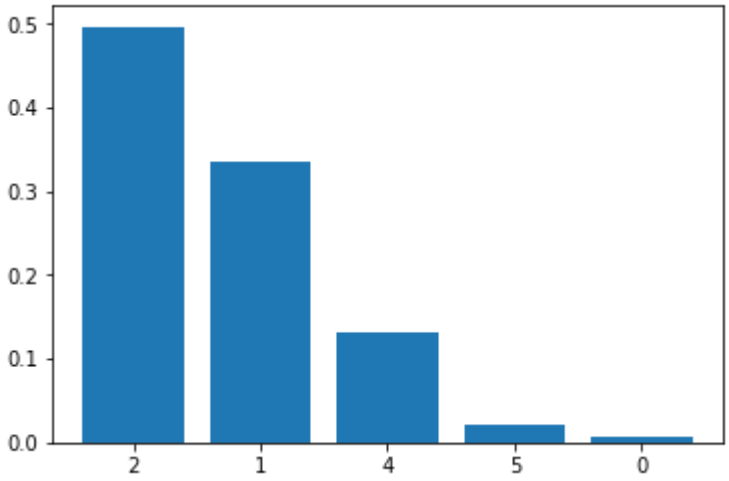

```
The kernel which activates/recognizes the shape: 20
```





Multiple kernel activations starting from 0

Activation for 20 kernel in 2 layer.



Probabilities of top 5 classes.

[5.5] Conv(32 -- 4x4) - Conv(64 -- 3x3) - MaxPool(2x2) - Conv(32 -- 4x4) - Dropout(0.75) - MaxPool(2x2) - Conv(32 -- 3x3) - Dense(256) - Dropout(0.5) - Dense(128) - Dropout(0.5)

In [0]:

```
epochs = 15
model = Sequential()
model.add(Conv2D(
    32, kernel_size=(4, 4), activation='relu', input_shape=input_shape
))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (4, 4), activation='relu'))
model.add(Dropout(rate=0.75))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(
    loss=keras.losses.categorical_crossentropy,
    optimizer=keras.optimizers.Adadelta(), metrics=['accuracy']
)

model.summary()

history = model.fit(
    x_train, y_train, batch_size=batch_size, epochs=epochs,
    verbose=1, validation_data=(x_test, y_test)
    #class_weight=class_weights
)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 61, 61, 32)        544
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 59, 59, 64)        18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 29, 29, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 26, 26, 32)        32800
_________________________________________________________________
dropout_11 (Dropout)         (None, 26, 26, 32)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 11, 11, 32)        9248
_________________________________________________________________
flatten_6 (Flatten)          (None, 3872)              0
_________________________________________________________________
dense_11 (Dense)             (None, 256)               991488
_________________________________________________________________
dropout_12 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_13 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_13 (Dense)             (None, 43)                5547
```

```
=================================================================
Total params: 1,091,019
Trainable params: 1,091,019
Non-trainable params: 0
_________________________________________________________________
Train on 27446 samples, validate on 11763 samples
Epoch 1/15
27446/27446 [==============================] - 10s 355us/step - loss: 2.8067
- acc: 0.2616 - val_loss: 1.3913 - val_acc: 0.7479
Epoch 2/15
27446/27446 [==============================] - 9s 323us/step - loss: 0.8320
- acc: 0.7555 - val_loss: 0.5685 - val_acc: 0.9355
Epoch 3/15
27446/27446 [==============================] - 9s 325us/step - loss: 0.4372
- acc: 0.8708 - val_loss: 0.2997 - val_acc: 0.9653
Epoch 4/15
27446/27446 [==============================] - 9s 327us/step - loss: 0.3041
- acc: 0.9099 - val_loss: 0.1937 - val_acc: 0.9773
Epoch 5/15
27446/27446 [==============================] - 9s 328us/step - loss: 0.2310
- acc: 0.9326 - val_loss: 0.1439 - val_acc: 0.9758
Epoch 6/15
27446/27446 [==============================] - 9s 326us/step - loss: 0.1861
- acc: 0.9444 - val_loss: 0.1127 - val_acc: 0.9823
Epoch 7/15
27446/27446 [==============================] - 9s 325us/step - loss: 0.1586
- acc: 0.9543 - val_loss: 0.1040 - val_acc: 0.9861
Epoch 8/15
27446/27446 [==============================] - 9s 325us/step - loss: 0.1280
- acc: 0.9624 - val_loss: 0.0869 - val_acc: 0.9870
Epoch 9/15
27446/27446 [==============================] - 9s 323us/step - loss: 0.1197
- acc: 0.9644 - val_loss: 0.0815 - val_acc: 0.9884
Epoch 10/15
27446/27446 [==============================] - 9s 323us/step - loss: 0.1001
- acc: 0.9698 - val_loss: 0.0566 - val_acc: 0.9890
Epoch 11/15
27446/27446 [==============================] - 9s 322us/step - loss: 0.0997
- acc: 0.9713 - val_loss: 0.0583 - val_acc: 0.9897
Epoch 12/15
27446/27446 [==============================] - 9s 322us/step - loss: 0.0868
- acc: 0.9735 - val_loss: 0.0536 - val_acc: 0.9905
Epoch 13/15
27446/27446 [==============================] - 9s 322us/step - loss: 0.0801
- acc: 0.9767 - val_loss: 0.0602 - val_acc: 0.9893
Epoch 14/15
27446/27446 [==============================] - 9s 323us/step - loss: 0.0735
- acc: 0.9784 - val_loss: 0.0508 - val_acc: 0.9901
Epoch 15/15
27446/27446 [==============================] - 9s 324us/step - loss: 0.0738
- acc: 0.9793 - val_loss: 0.0444 - val_acc: 0.9903
```

In [0]:

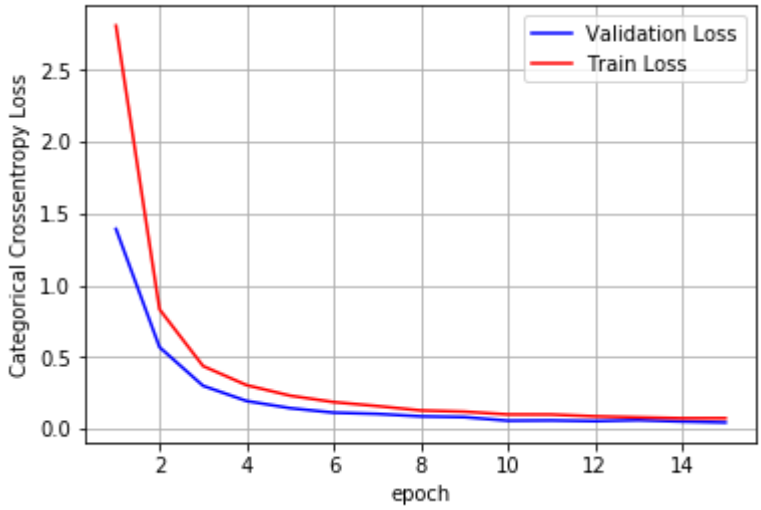
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(fig, x, vy, ty, ax)
```

```
Test score: 0.0443572680131209
Test accuracy: 0.9903085947462382
```



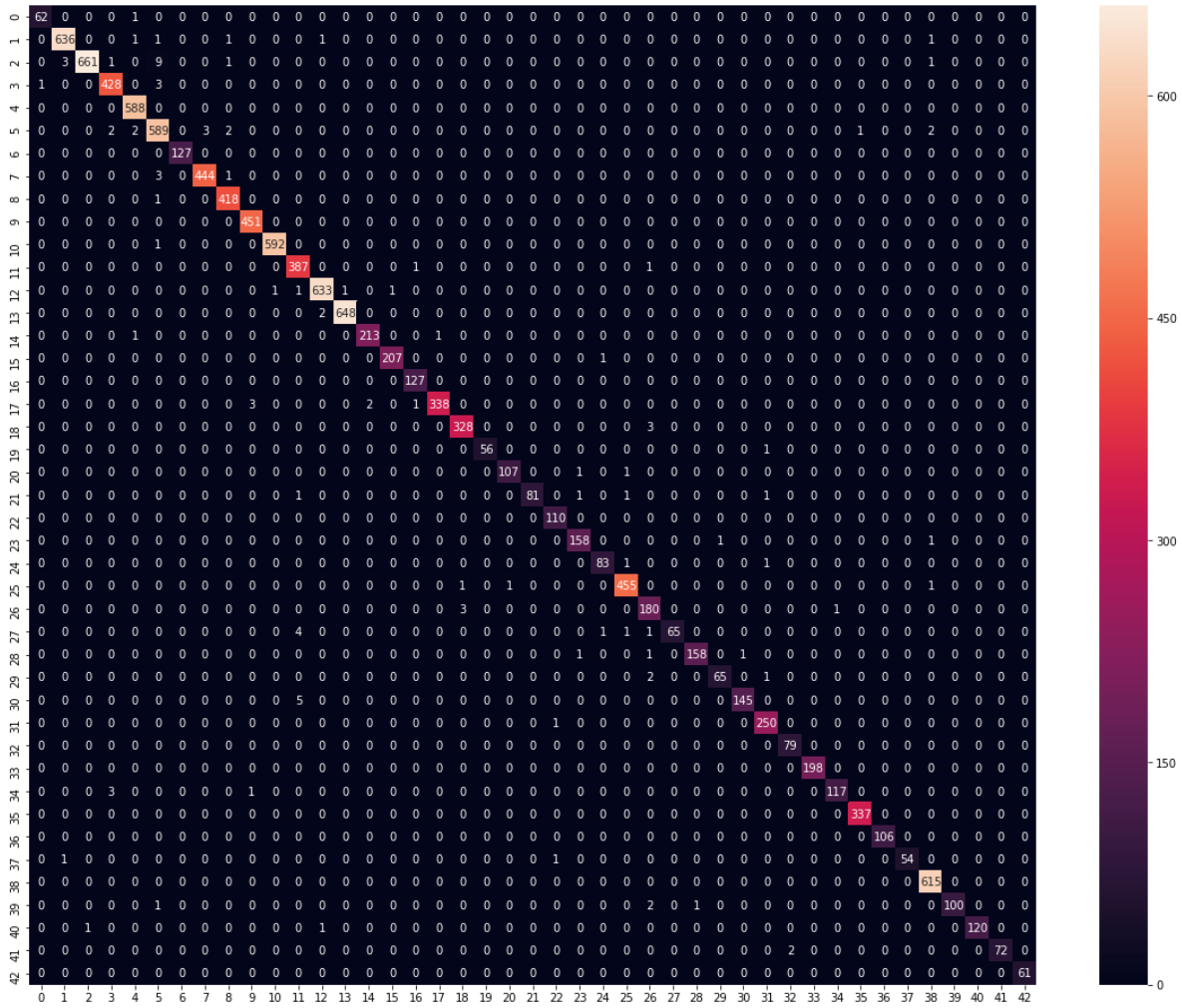
In [0]:

```
plt.figure(figsize=(20,16))
y_prediction = model.predict(x_test)
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_prediction, axis = 1)
# Convert validation observations to one hot vectors
y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
print("-----------------------------------------------------------")
print("Micro F1 score", f1_score(y_true, y_pred_classes, average='micro'))
print("-----------------------------------------------------------")
confusion_mtx = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(confusion_mtx, annot=True, fmt="d")
```

```
-----------------------------------------------------------
Micro F1 score 0.9903085947462382
-----------------------------------------------------------
```

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b00127978>
```



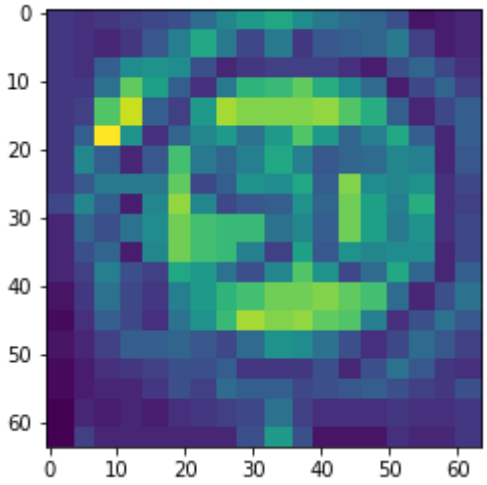
In [0]:

```
layer_outputs = [layer.output for layer in model.layers]
activation_model = Model(inputs=model.input, outputs=layer_outputs)
```

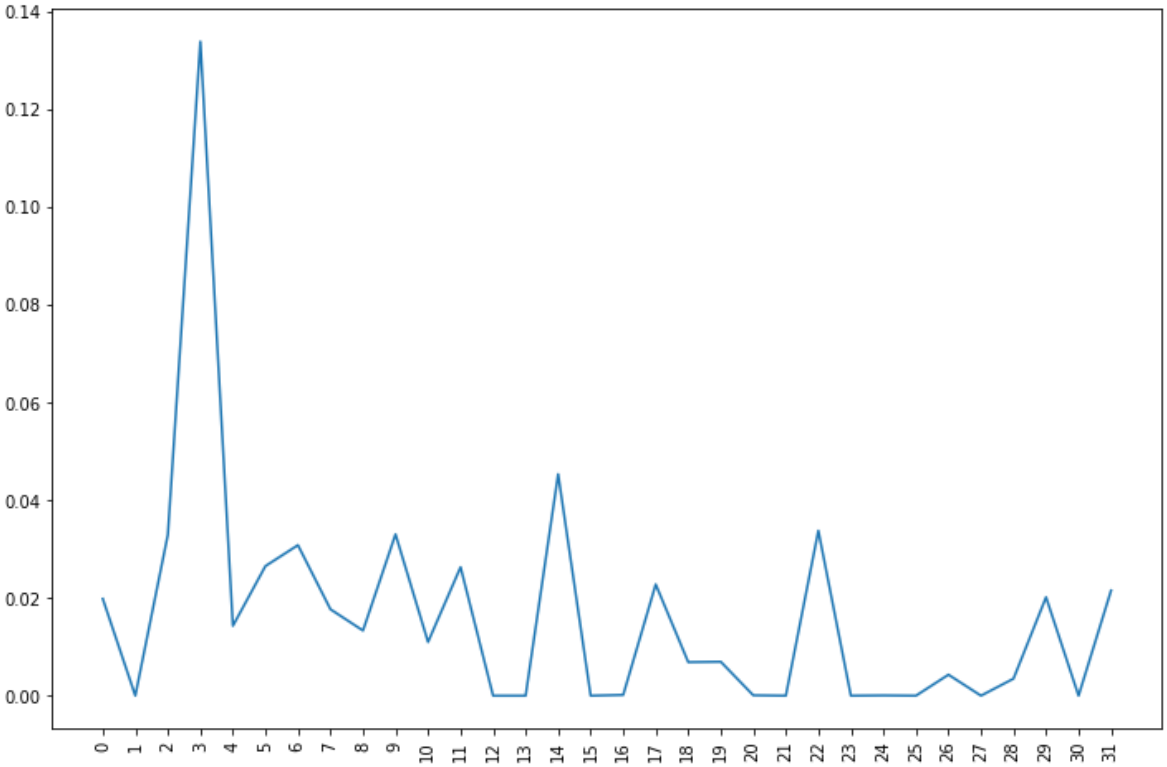
In [0]:

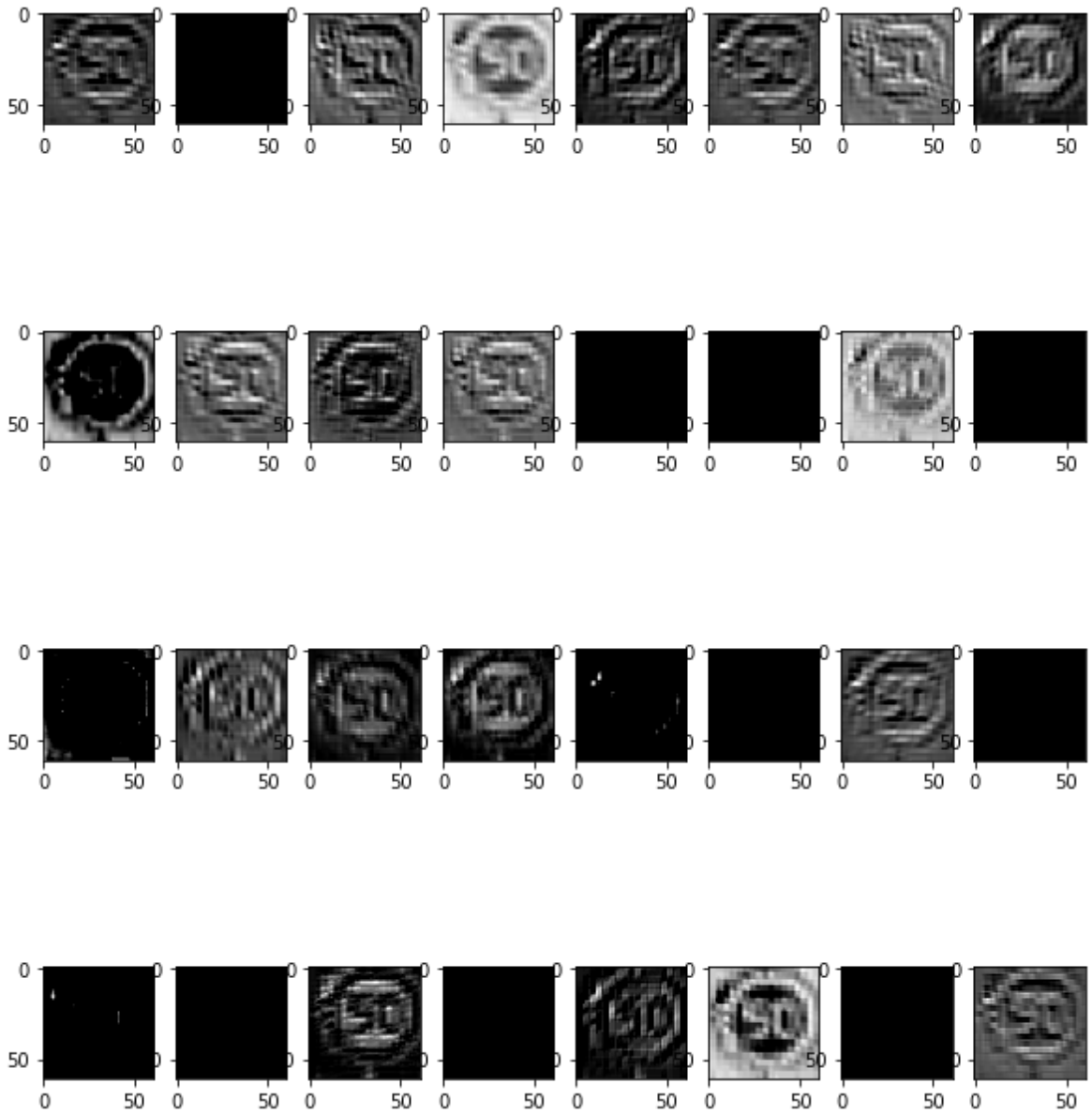
```
idx = 11078
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

```
Actual class it belongs to: 2
Predicted class 2
Actual training image
```

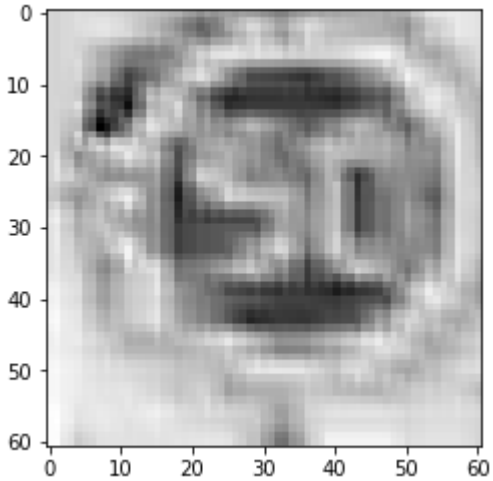



```
The kernel which activates/recognizes the shape: 3
```

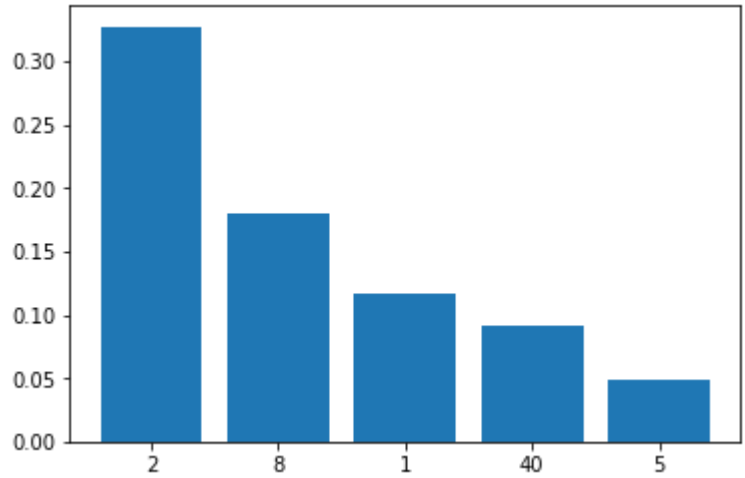




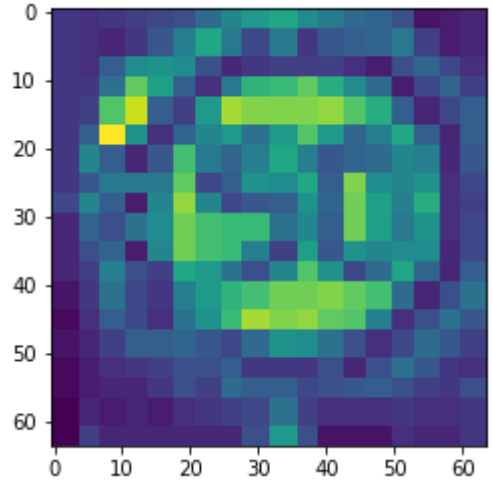
Multiple kernel activations starting from 0



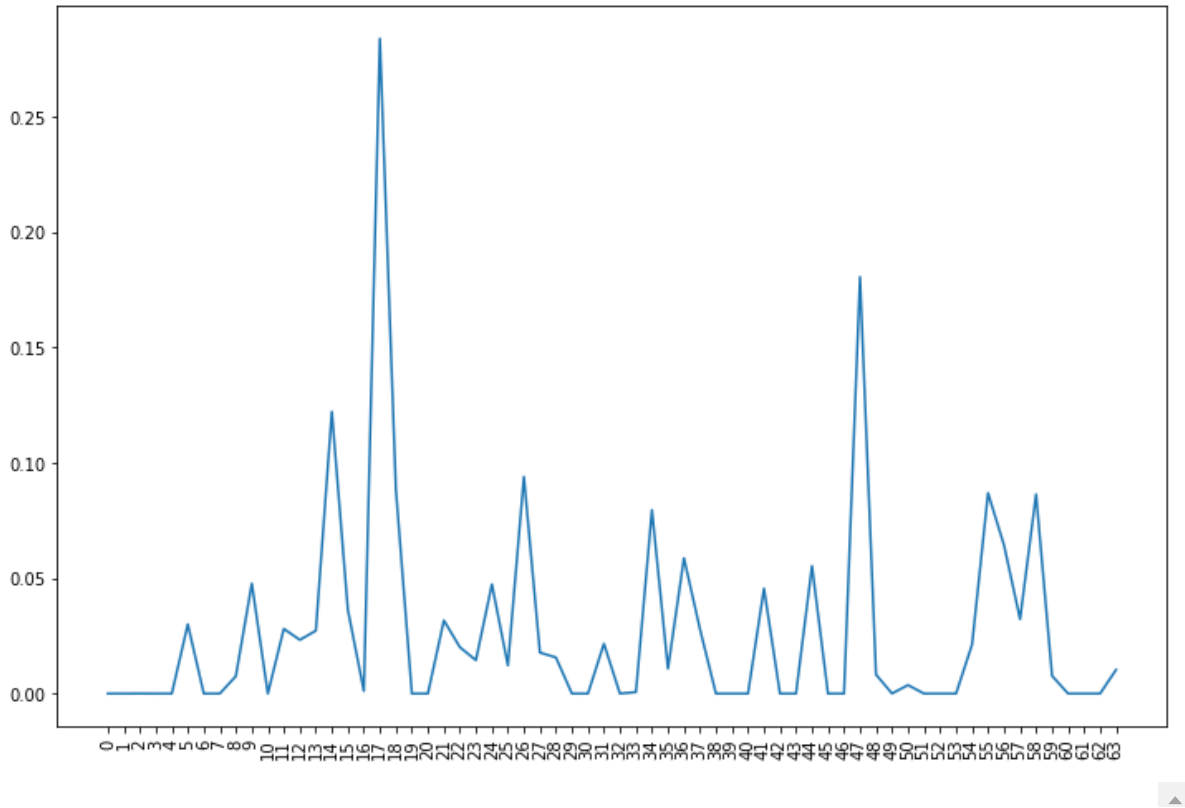
Activation for 3 kernel in 0 layer.

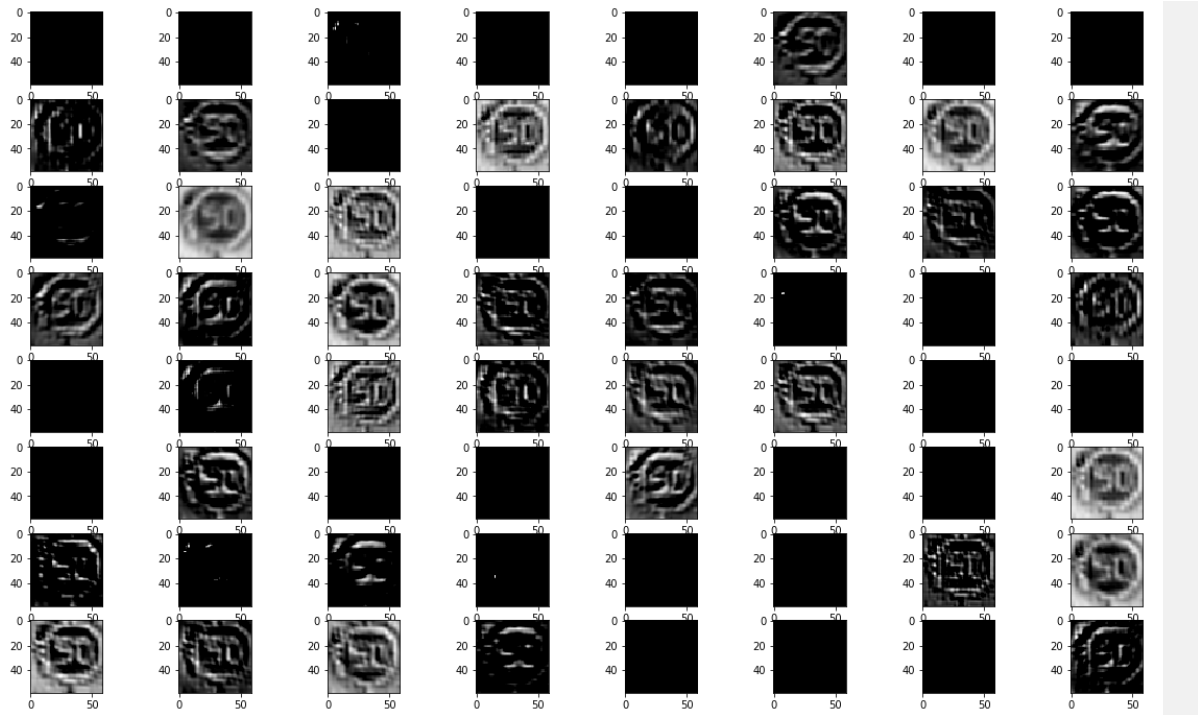



```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 2
Actual training image
```

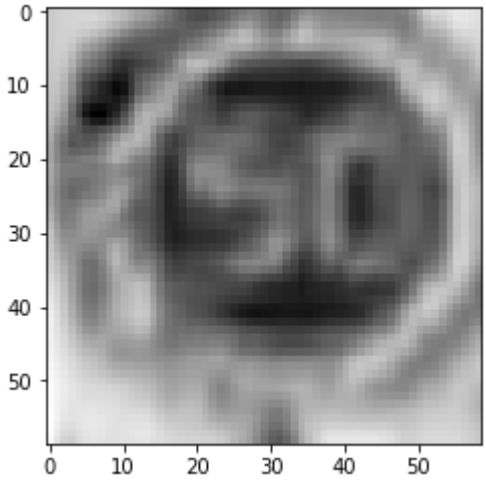



```
The kernel which activates/recognizes the shape: 17
```

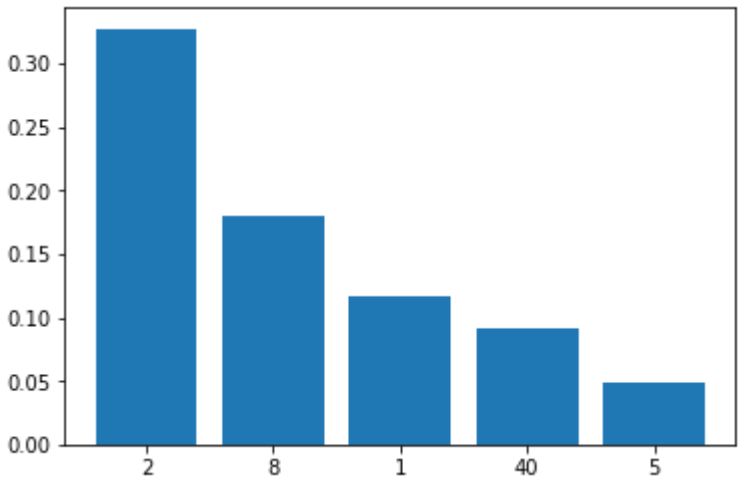




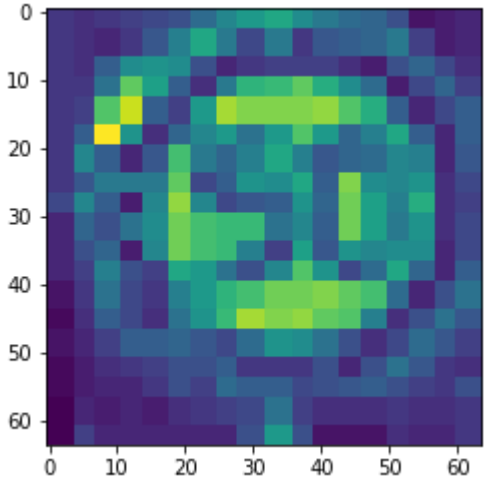
Multiple kernel activations starting from 0



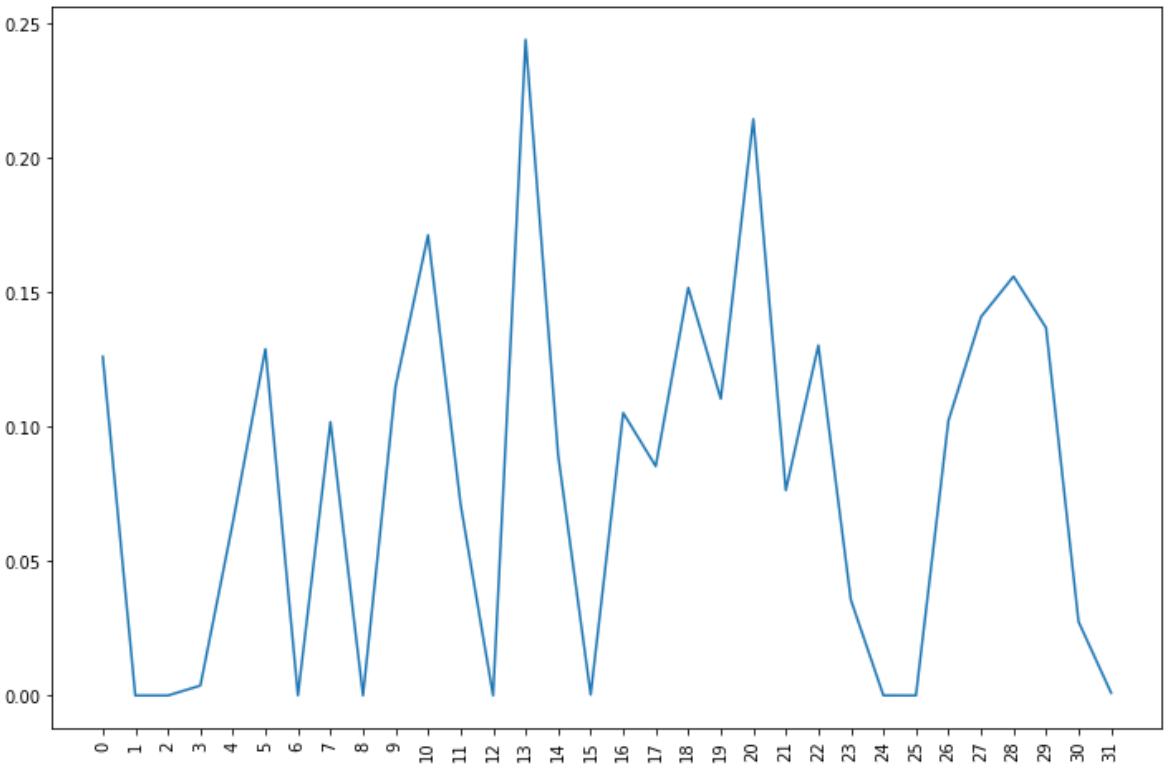
Activation for 17 kernel in 1 layer.



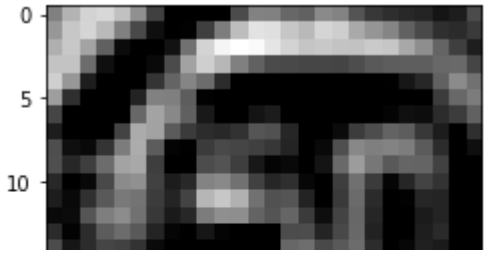
```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 2
Actual training image
```



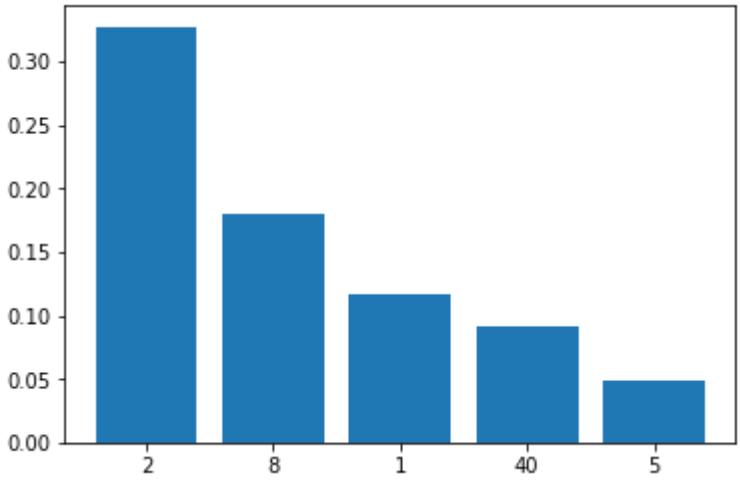

```
The kernel which activates/recognizes the shape: 13
```



Multiple kernel activations starting from 0



Activation for 13 kernel in 3 layer.

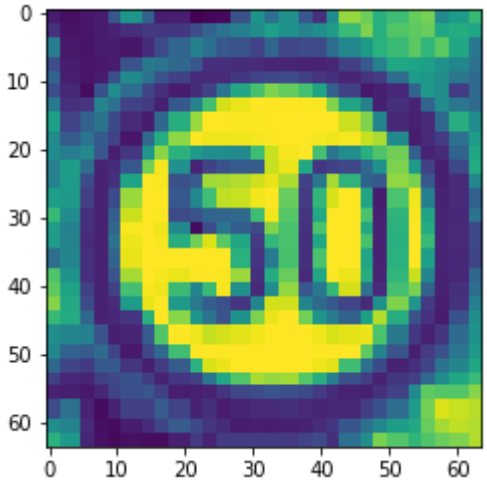


Probabilities of top 5 classes.

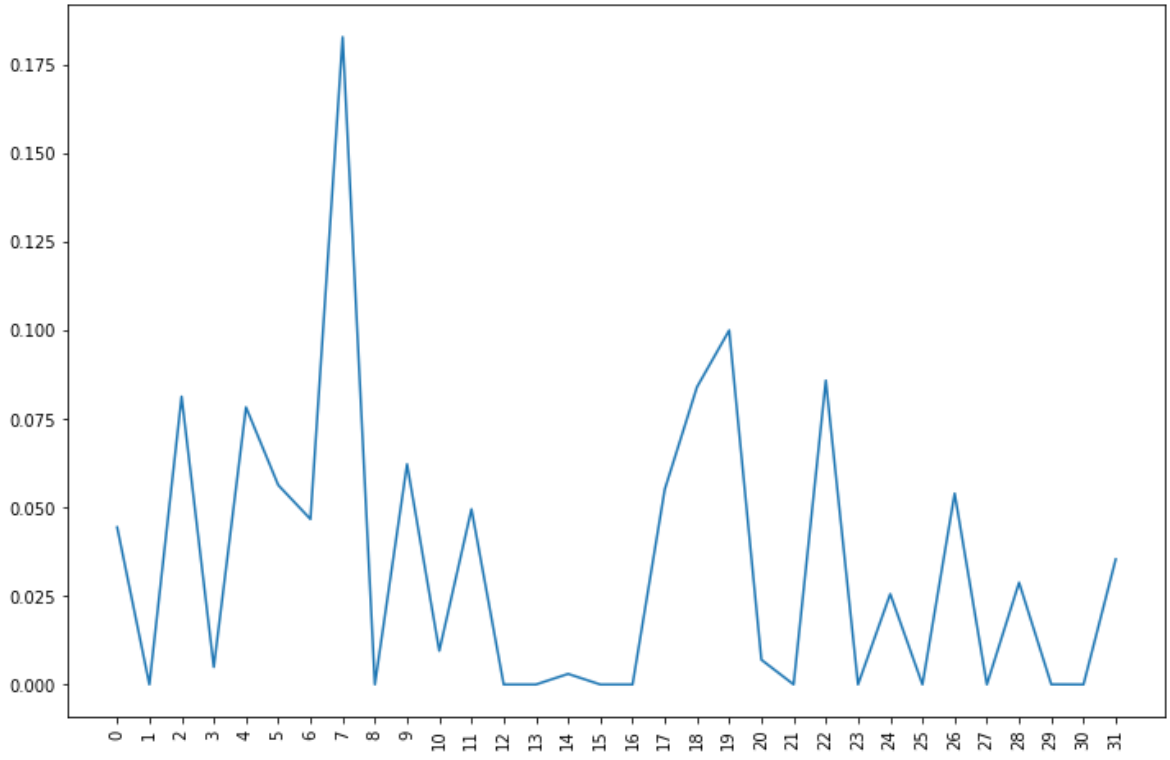
In [0]:

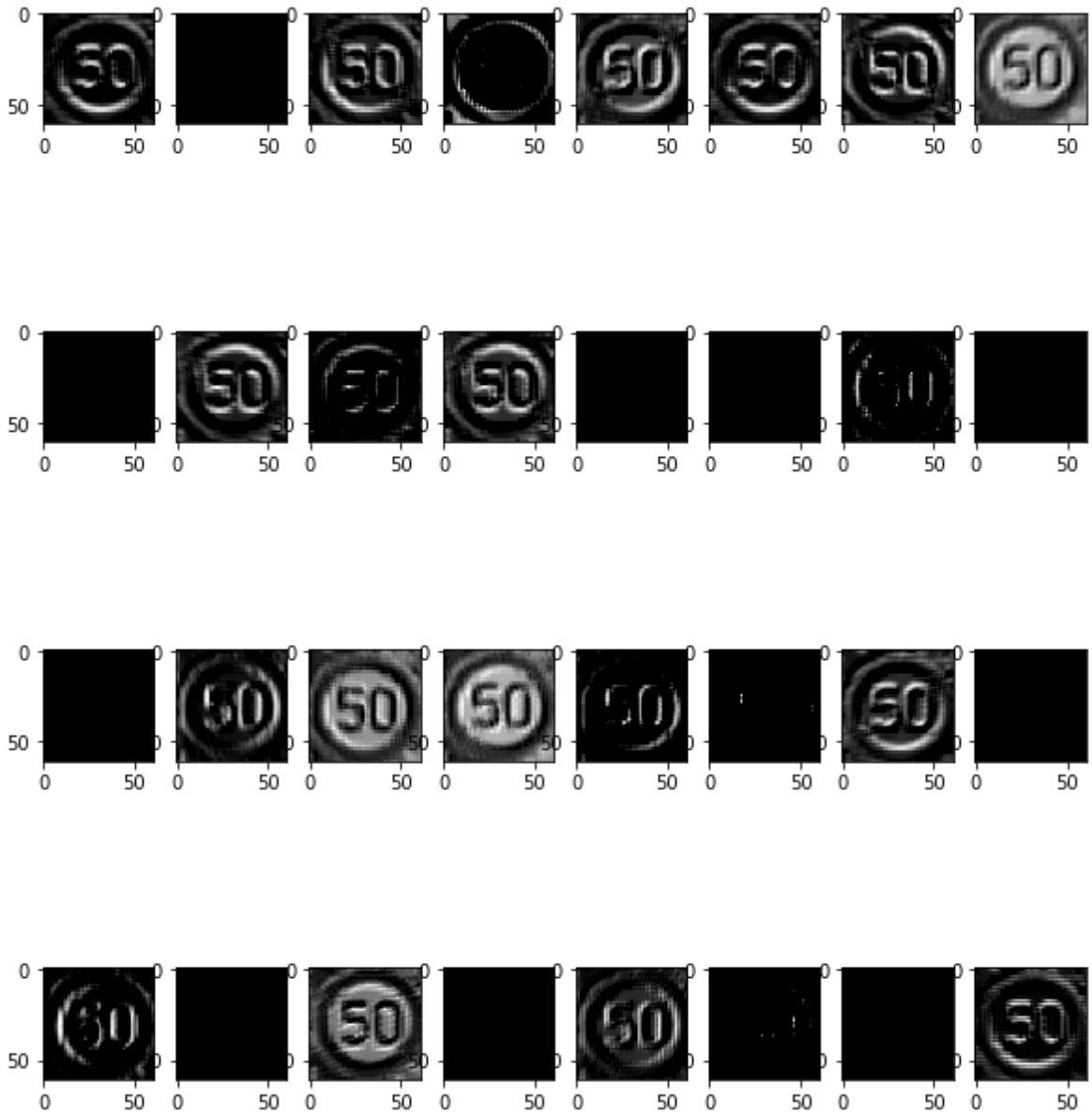
```
idx = 9777
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

```
Actual class it belongs to: 2
Predicted class 2
Actual training image
```

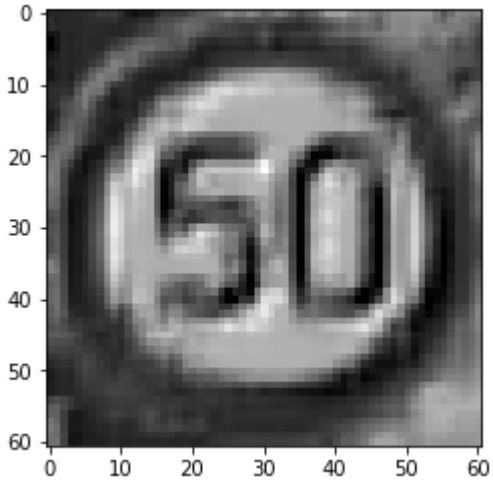


```
The kernel which activates/recognizes the shape: 7
```

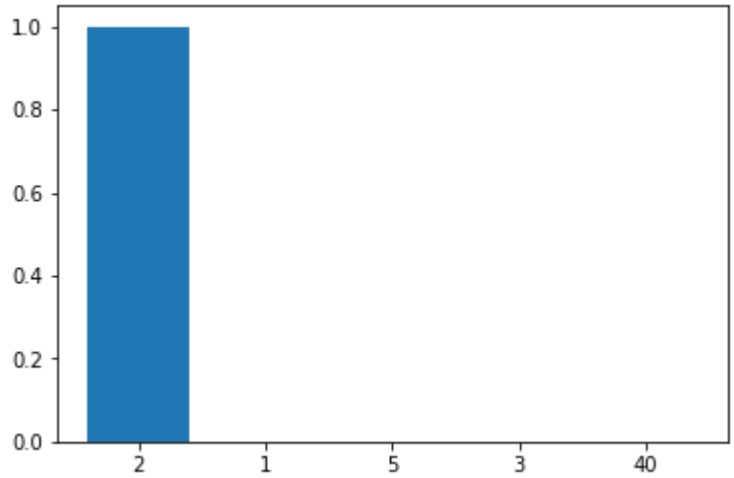




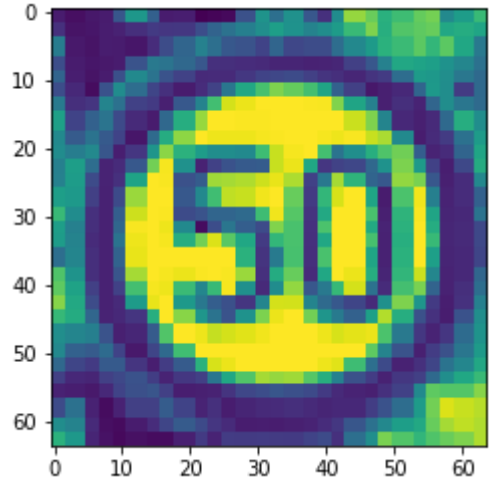
Multiple kernel activations starting from 0



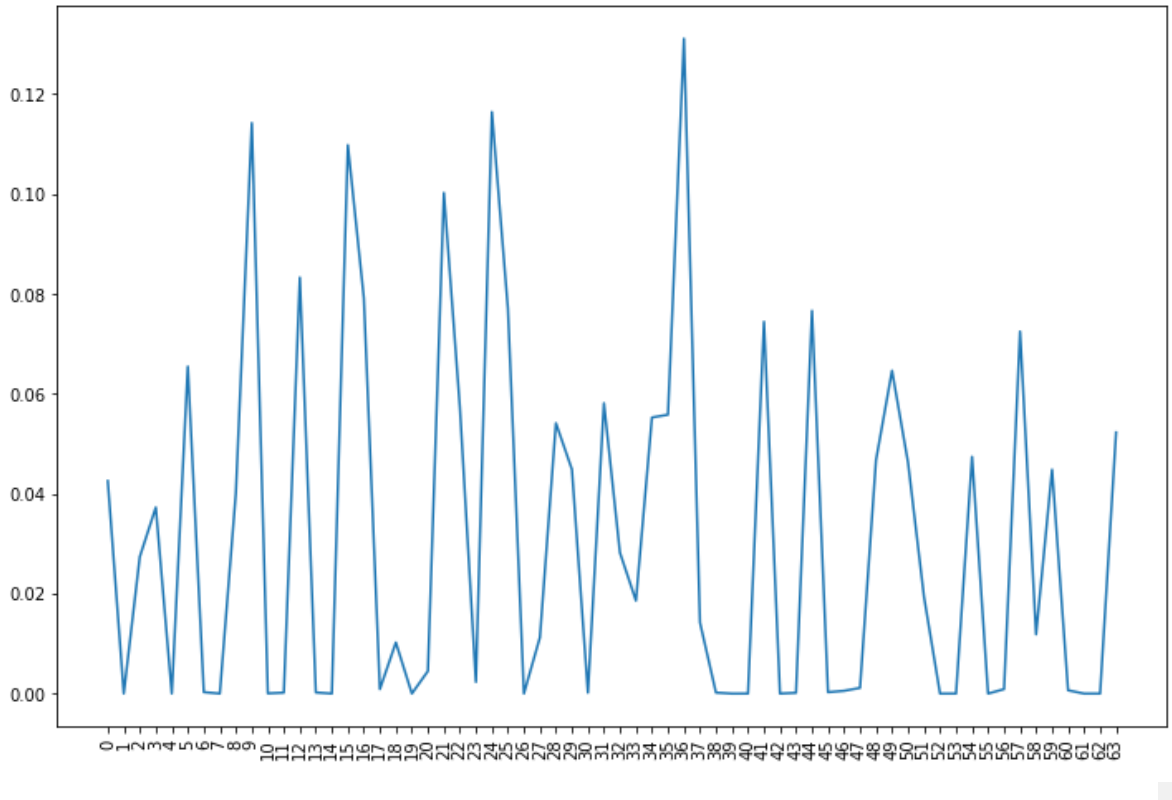
Activation for 7 kernel in 0 layer.

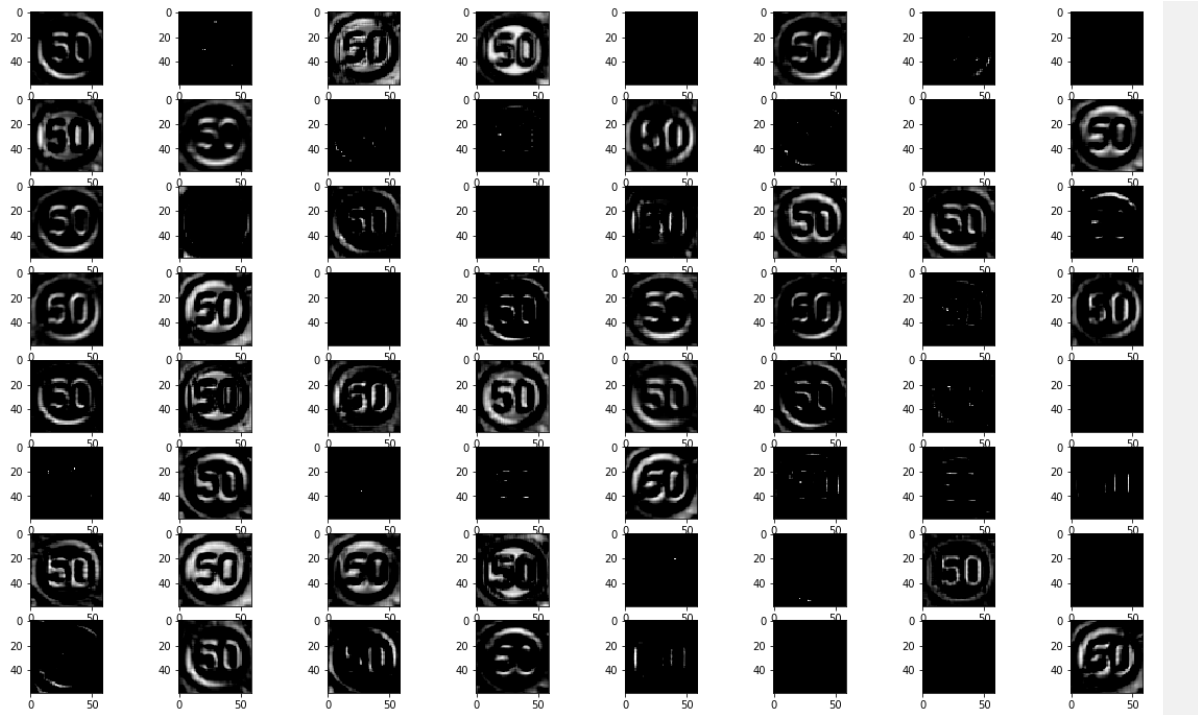



```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 2
Actual training image
```

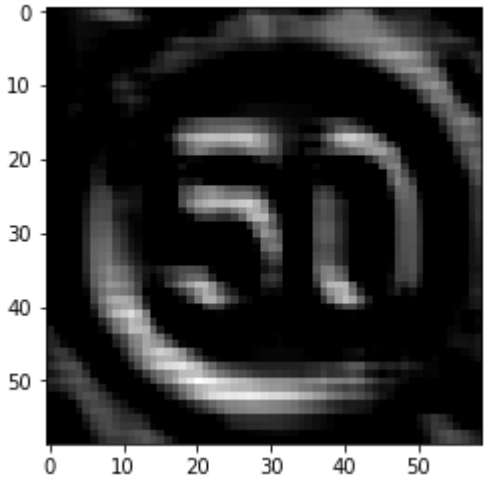



```
The kernel which activates/recognizes the shape: 36
```

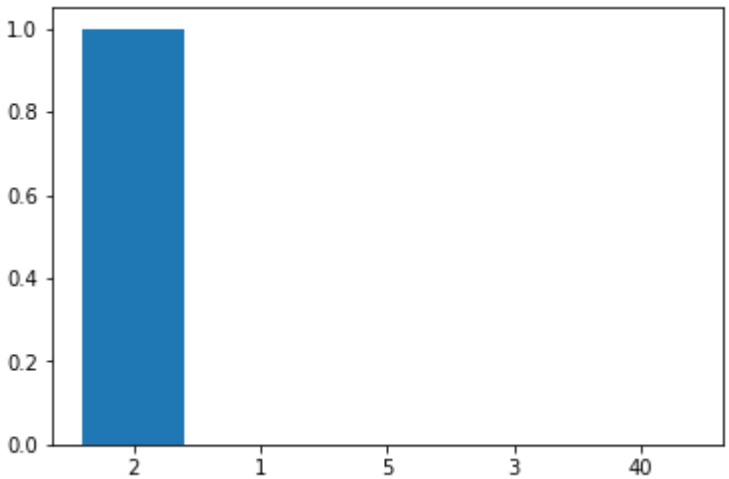




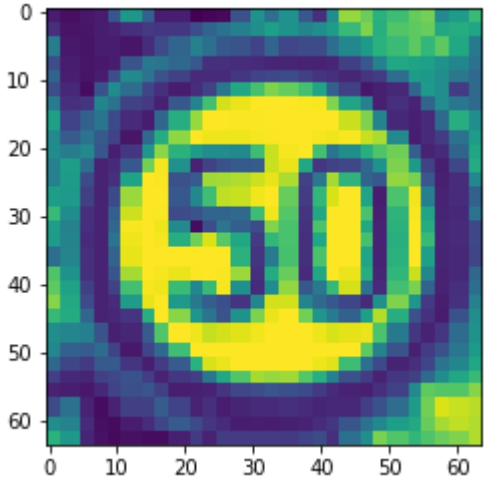
Multiple kernel activations starting from 0



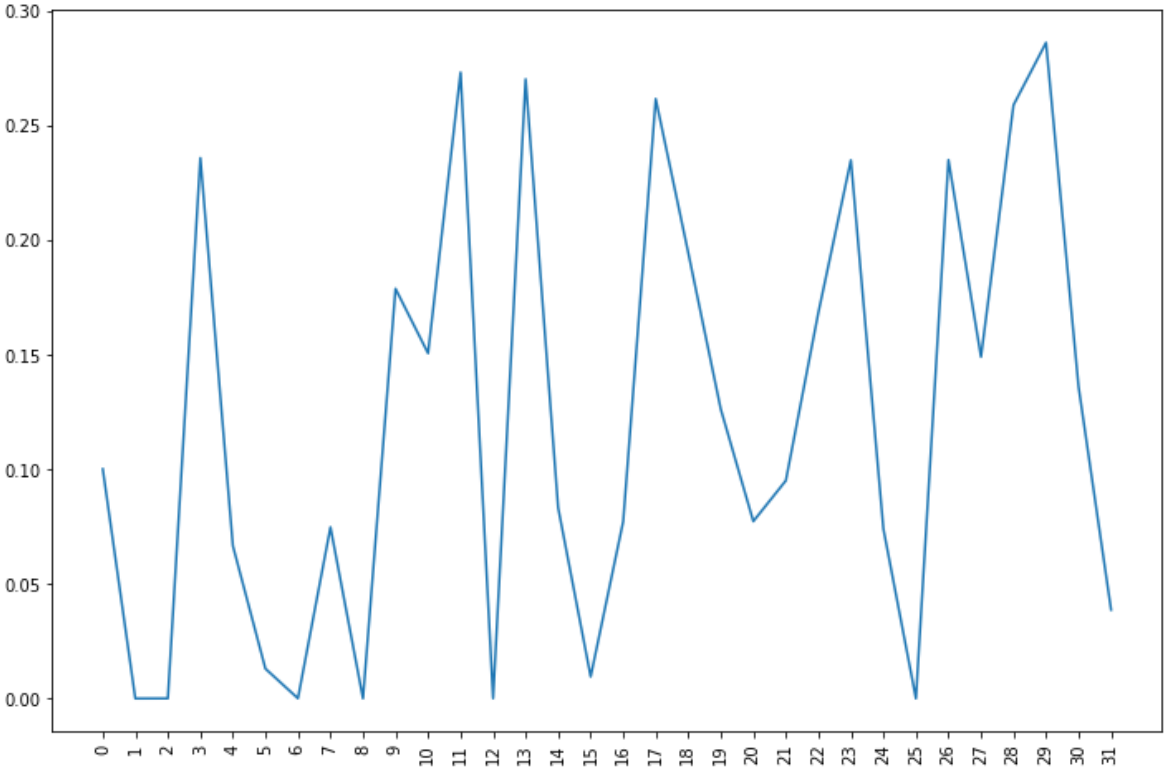
Activation for 36 kernel in 1 layer.

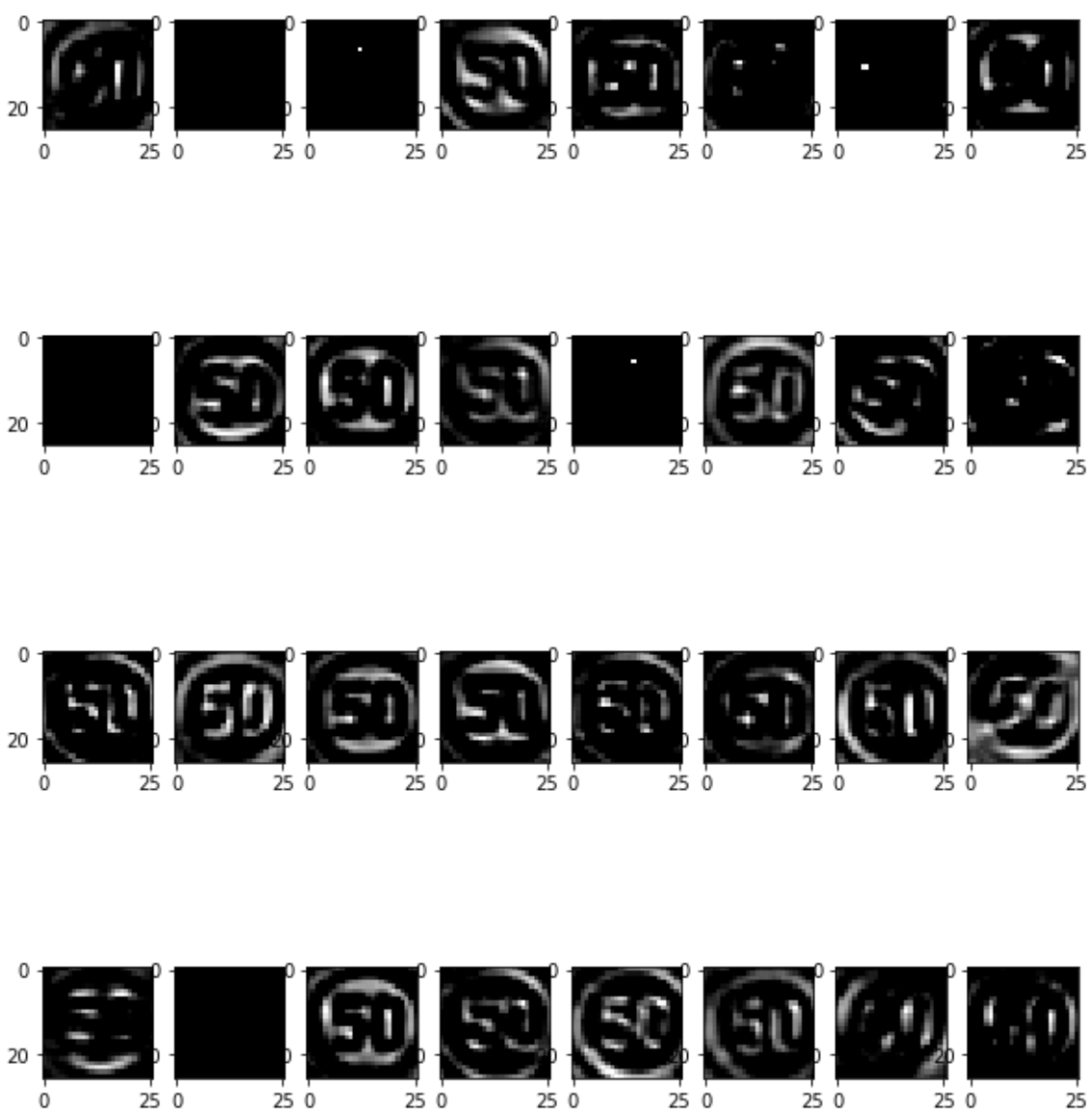


```
Probabilities of top 5 classes.
Actual class it belongs to: 2
Predicted class 2
Actual training image
```

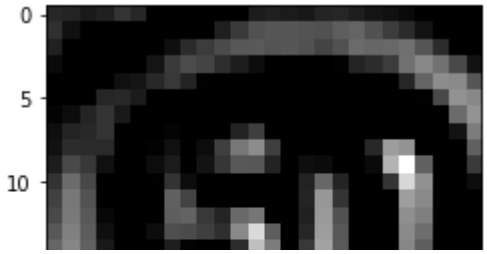


```
The kernel which activates/recognizes the shape: 29
```

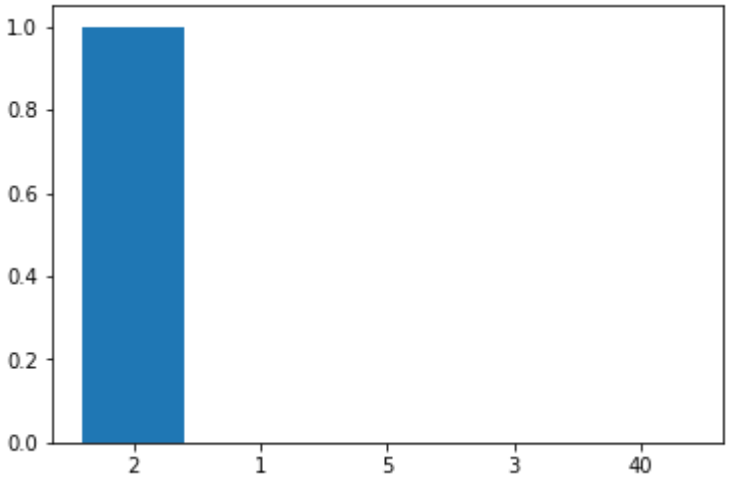




Multiple kernel activations starting from 0



Activation for 29 kernel in 3 layer.

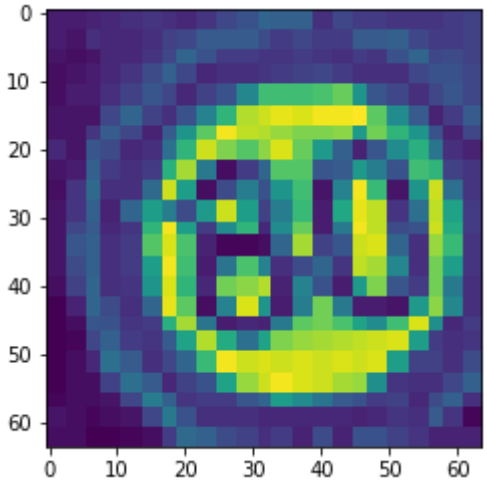


Probabilities of top 5 classes.

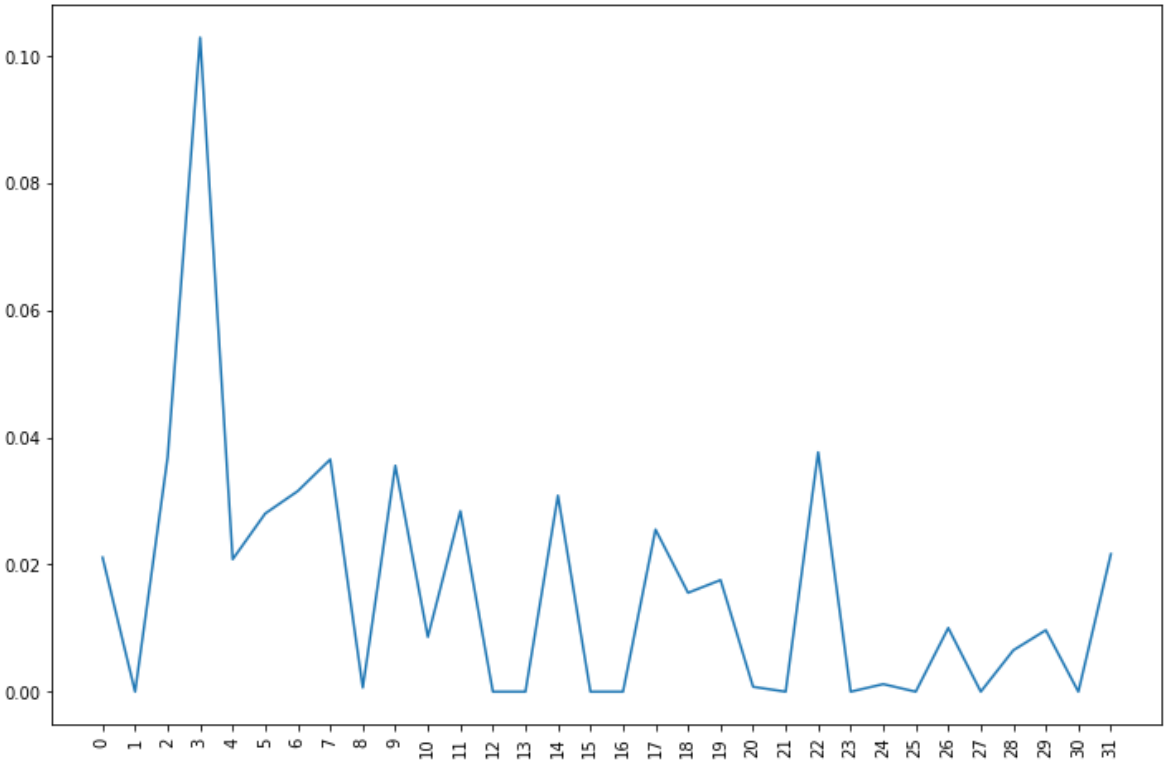
In [0]:

```
idx = 11650
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

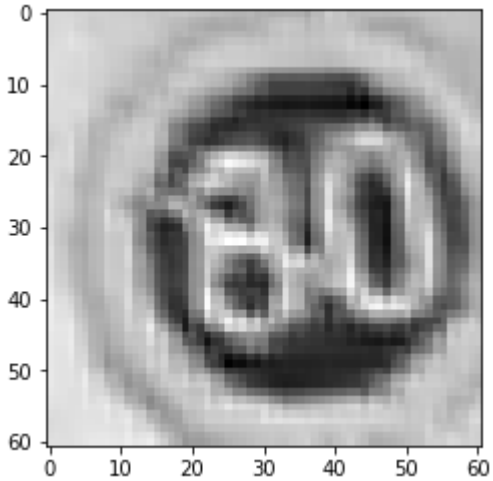
```
Actual class it belongs to: 5
Predicted class 5
Actual training image
```



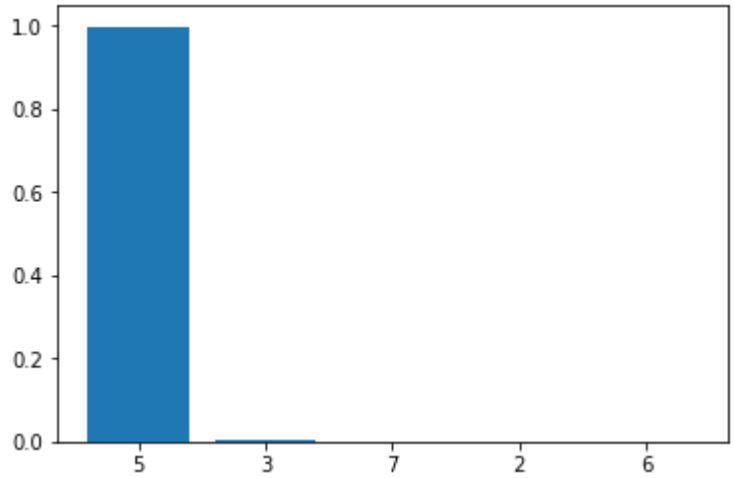

```
The kernel which activates/recognizes the shape: 3
```



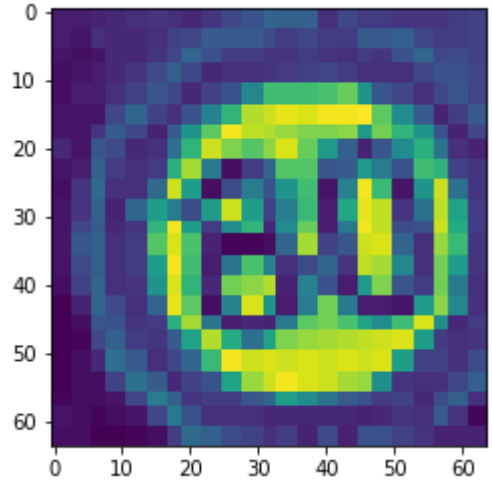
Multiple kernel activations starting from 0



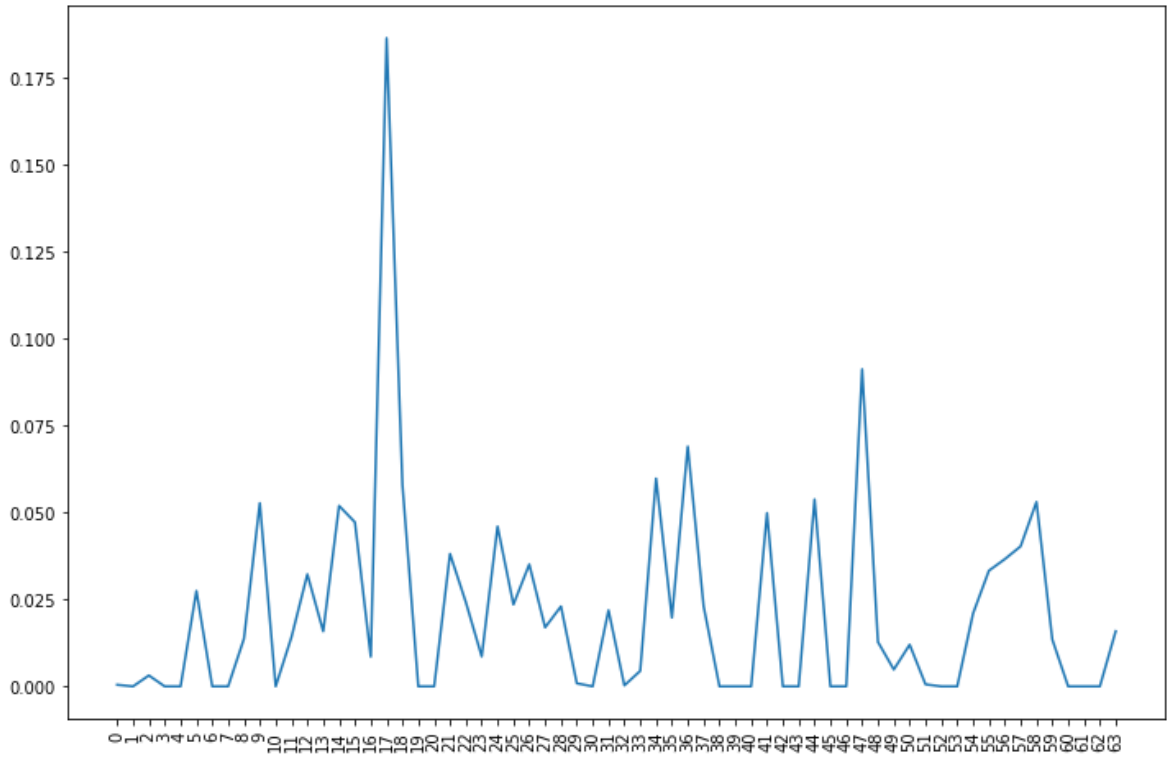
Activation for 3 kernel in 0 layer.

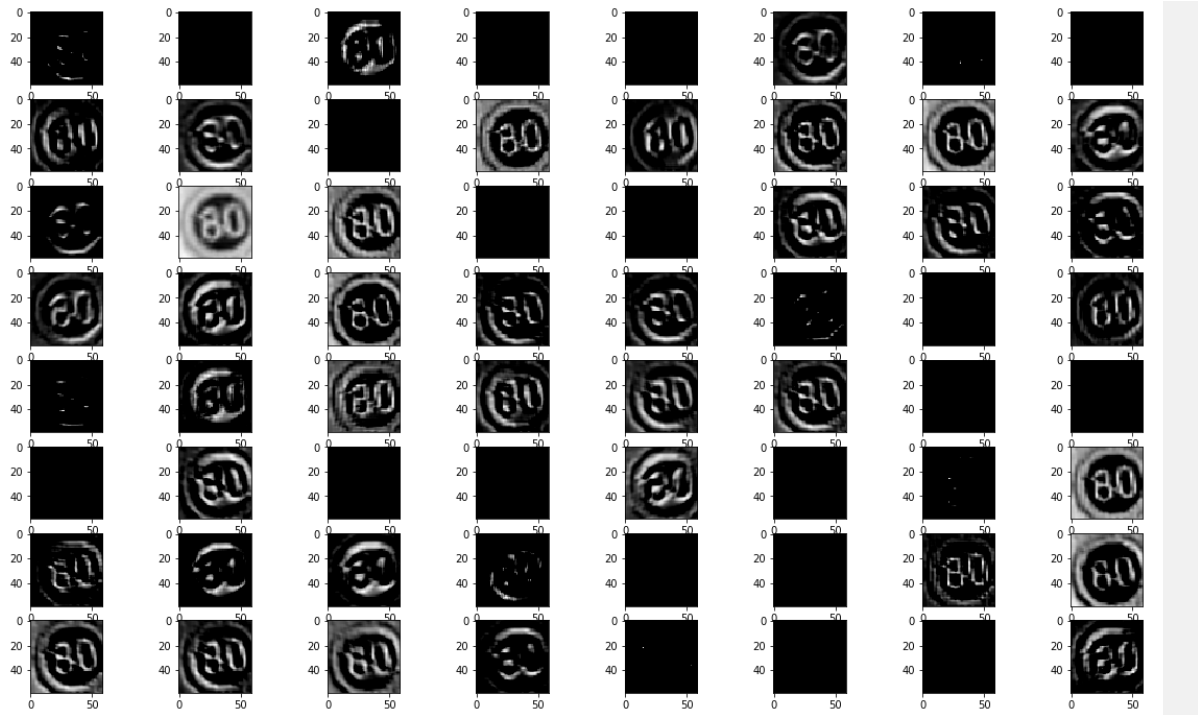



```
Probabilities of top 5 classes.
Actual class it belongs to: 5
Predicted class 5
Actual training image
```

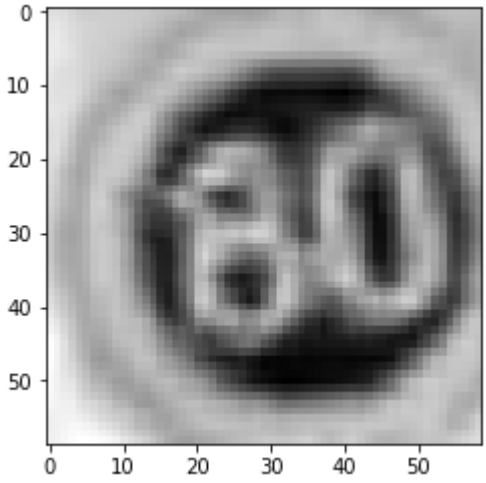



```
The kernel which activates/recognizes the shape: 17
```

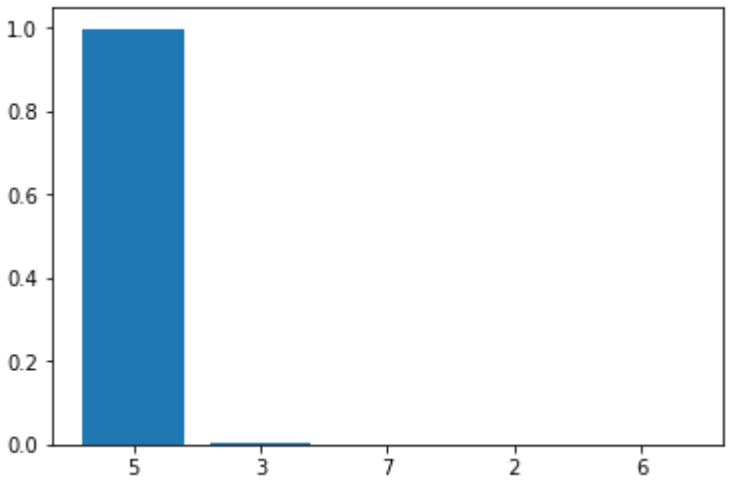




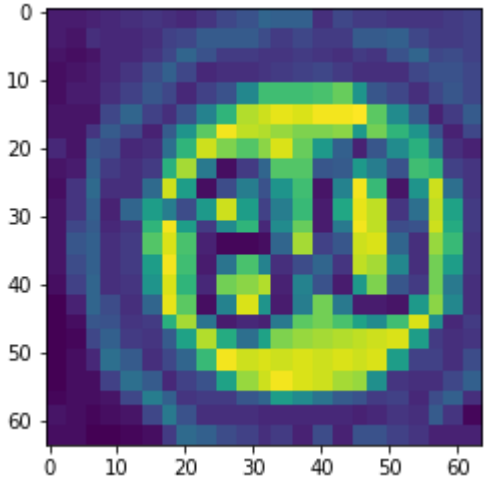
Multiple kernel activations starting from 0



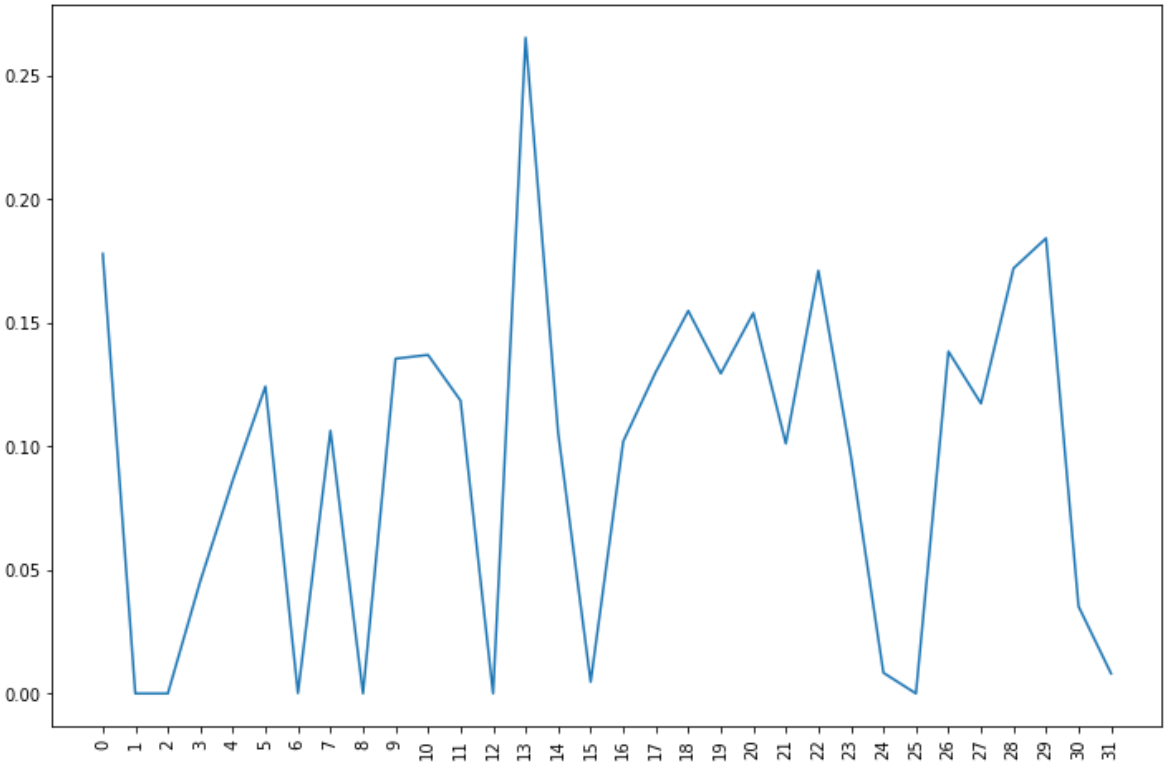
Activation for 17 kernel in 1 layer.



```
Probabilities of top 5 classes.
Actual class it belongs to: 5
Predicted class 5
Actual training image
```

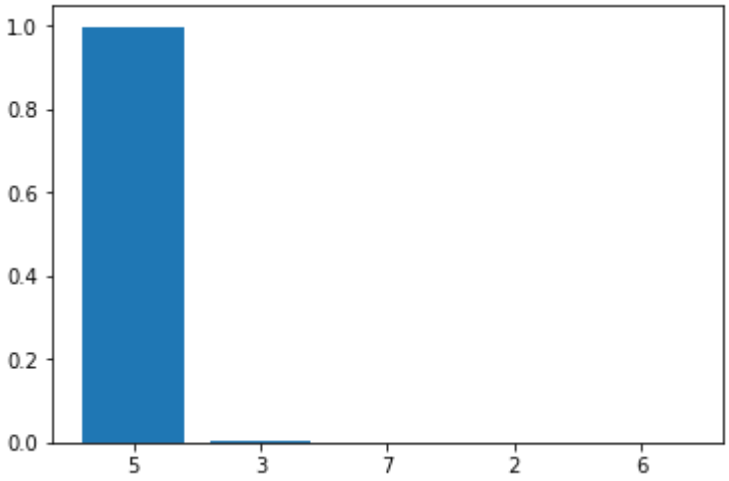



```
The kernel which activates/recognizes the shape: 13
```



Multiple kernel activations starting from 0

Activation for 13 kernel in 3 layer.



Probabilities of top 5 classes.

In [0]:

```
actual_class = []
index_image = []
predicted_class = []
for idx, im in enumerate(y_true):
    if im != np.argmax(y_prediction[idx]):
        actual_class.append(im)
        predicted_class.append(np.argmax(y_prediction[idx]))
        index_image.append(idx)
np.array(index_image)
```

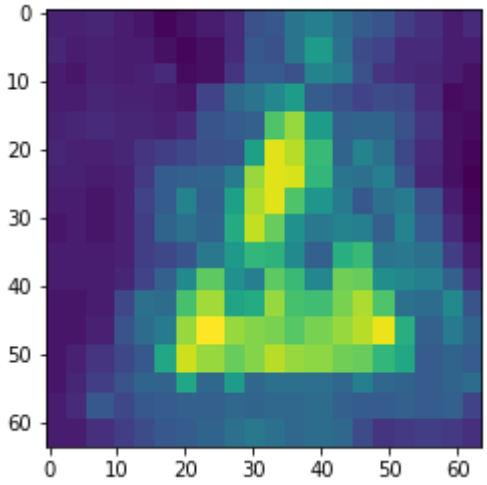
Out[104]:

```
array([   37,    41,   151,   365,   569,   572,   592,   621,   654,
         696,   836,   972,   981,  1193,  1204,  1210,  1321,  1395,
        1518,  1593,  1748,  1750,  1880,  1915,  1926,  2165,  2288,
        2325,  2365,  2392,  2639,  2925,  2967,  2978,  3080,  3450,
        3510,  3592,  3669,  3823,  3853,  3950,  4083,  4316,  4558,
        4738,  4854,  4926,  5032,  5089,  5189,  5240,  5491,  5513,
        5618,  5666,  5995,  6102,  6166,  6175,  6806,  6930,  7086,
        7210,  7269,  7378,  7481,  7520,  7521,  7533,  7570,  7626,
        7678,  7700,  7716,  7796,  7914,  8200,  8207,  8235,  8346,
        8348,  8412,  8492,  8510,  8578,  8598,  8632,  8723,  8735,
        8739,  9361,  9664,  9878, 10073, 10125, 10589, 10621, 10651,
       10814, 10845, 10917, 11042, 11090, 11211, 11301, 11307, 11333,
       11361, 11366, 11527, 11653, 11697, 11711])
```

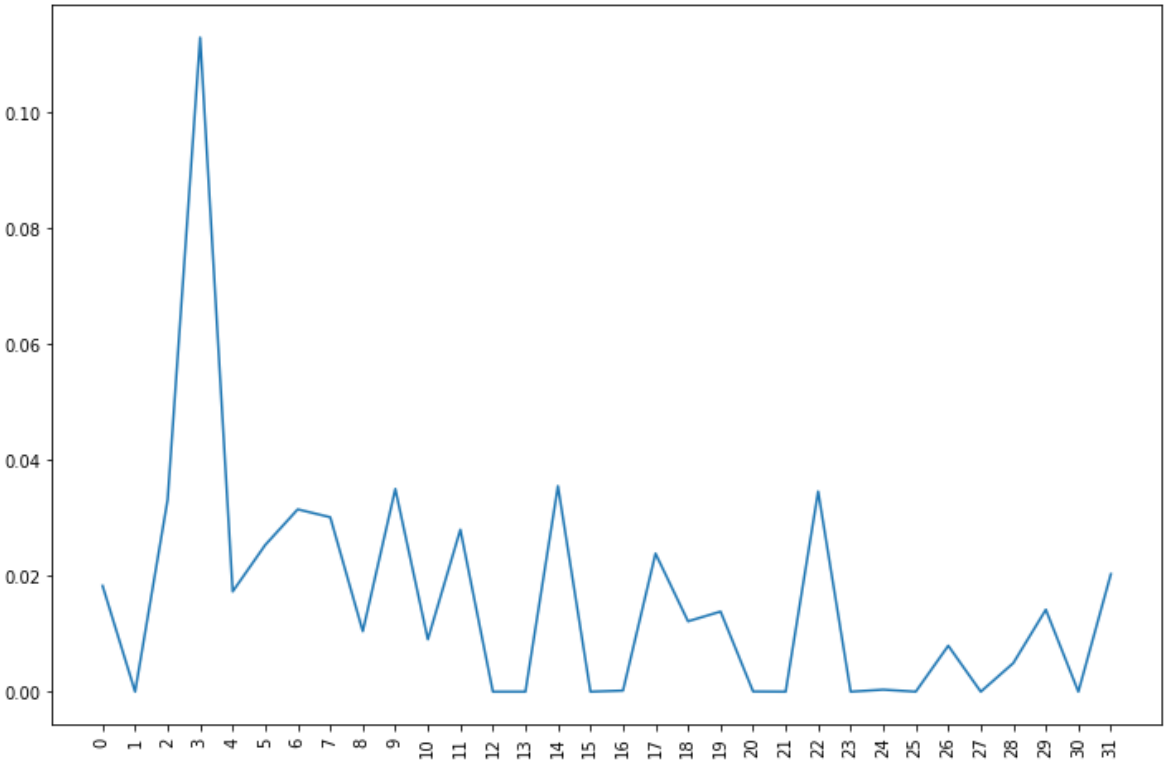
In [0]:

```
idx = 7210
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

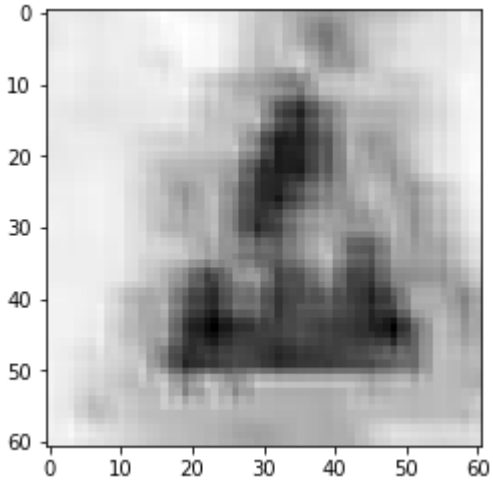
```
Actual class it belongs to: 28
Predicted class 23
Actual training image
```



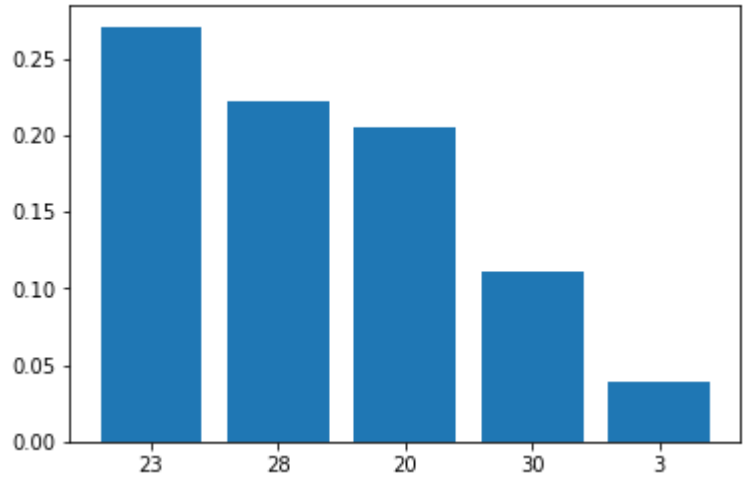

```
The kernel which activates/recognizes the shape: 3
```



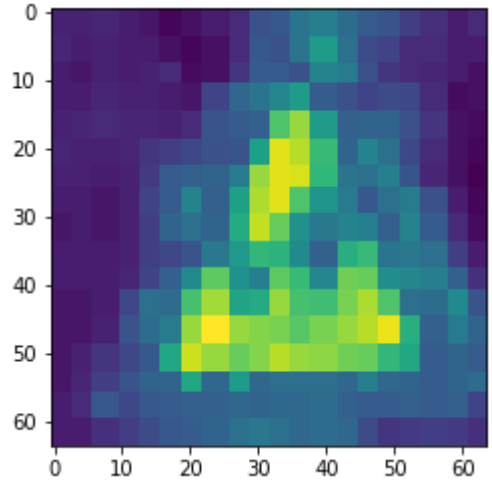
Multiple kernel activations starting from 0



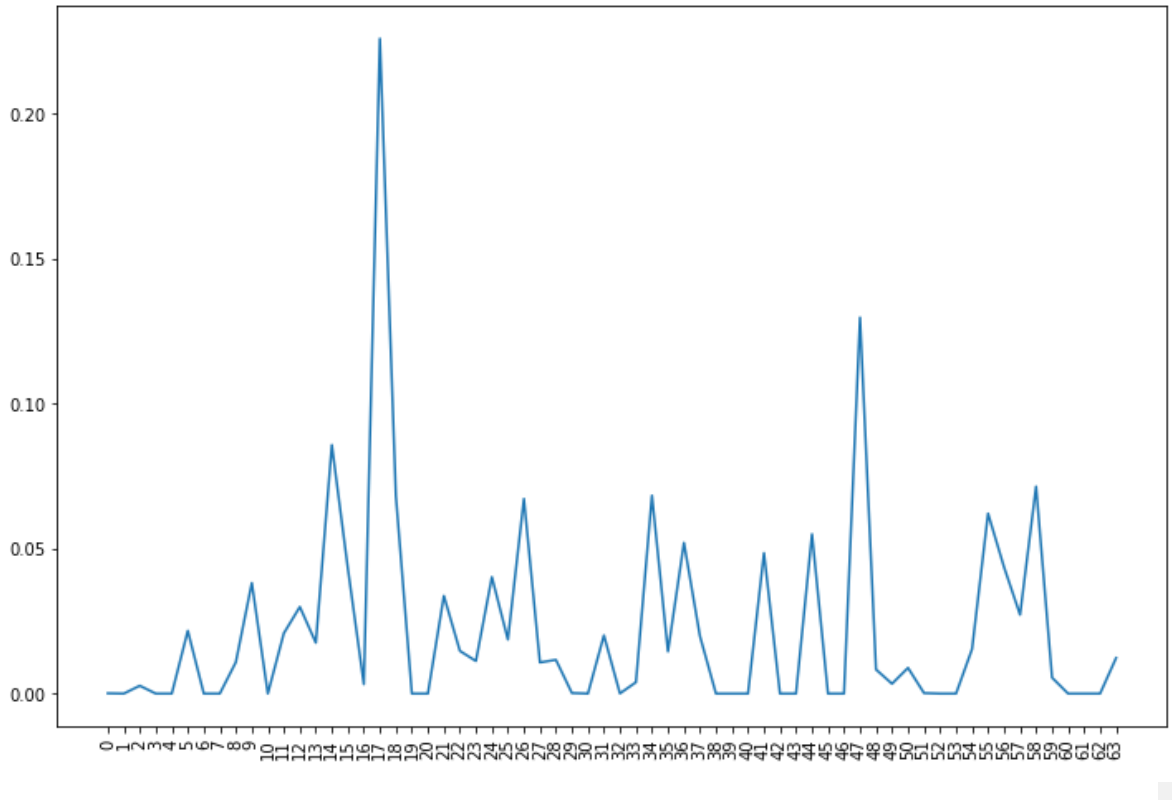
Activation for 3 kernel in 0 layer.

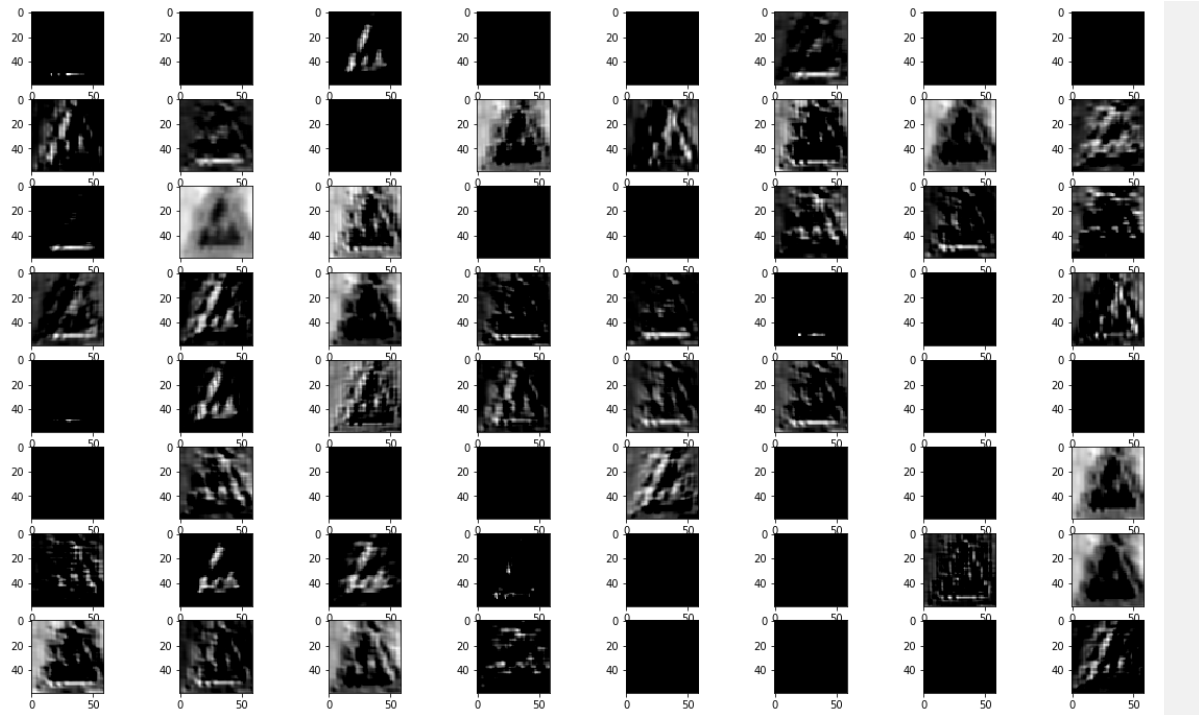



```
Probabilities of top 5 classes.
Actual class it belongs to: 28
Predicted class 23
Actual training image
```

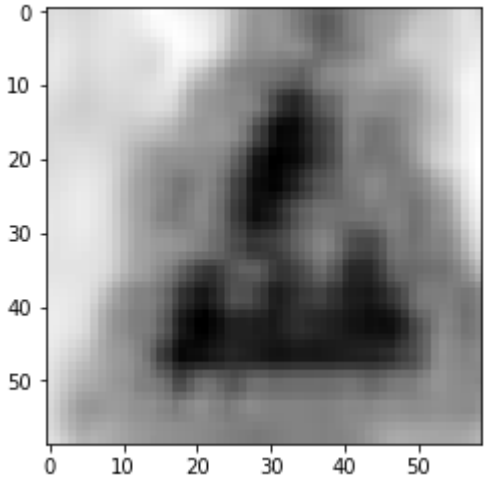



```
The kernel which activates/recognizes the shape: 17
```

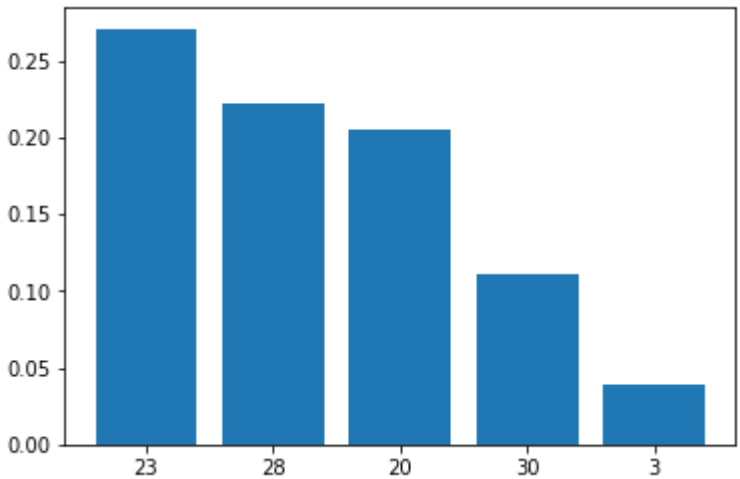




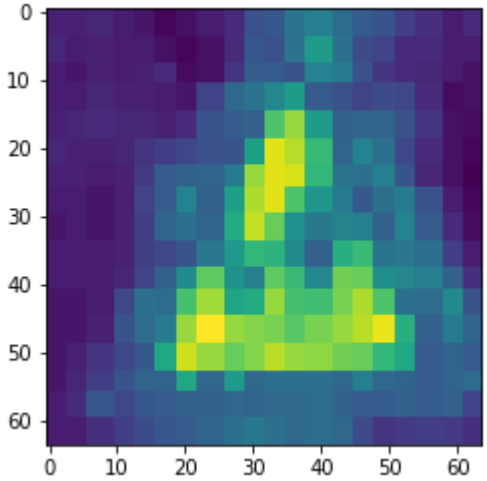
Multiple kernel activations starting from 0



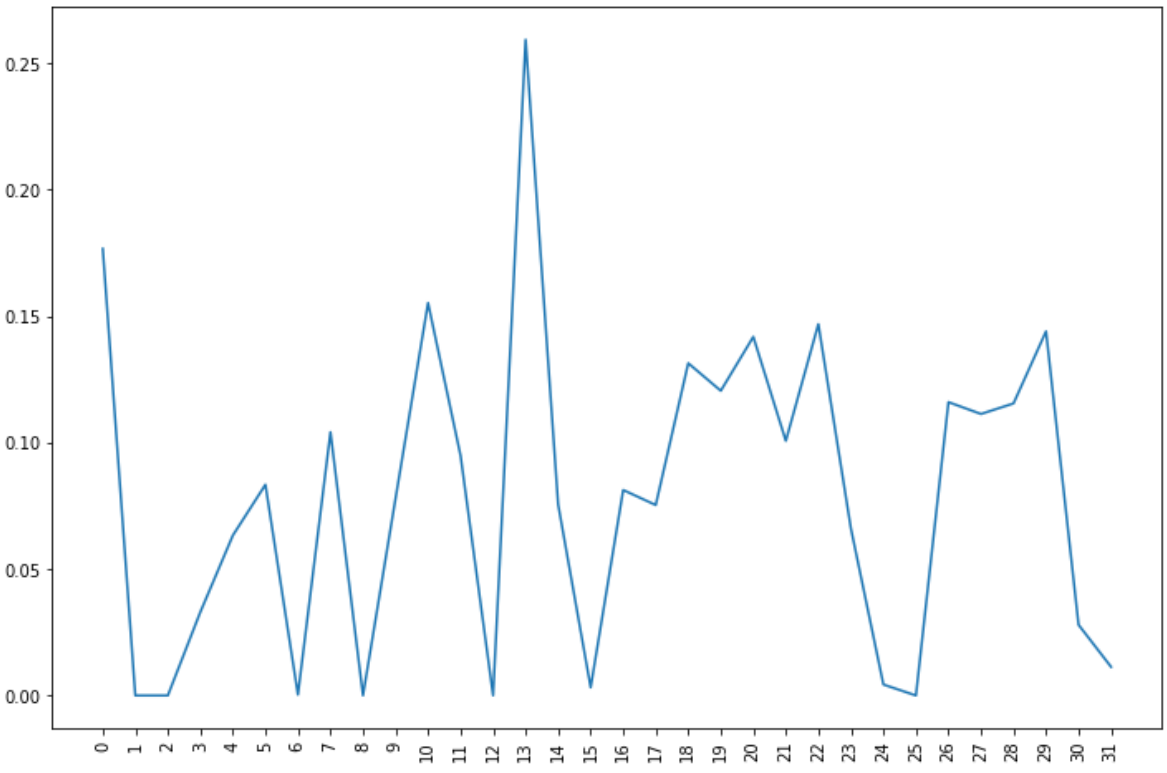
Activation for 17 kernel in 1 layer.

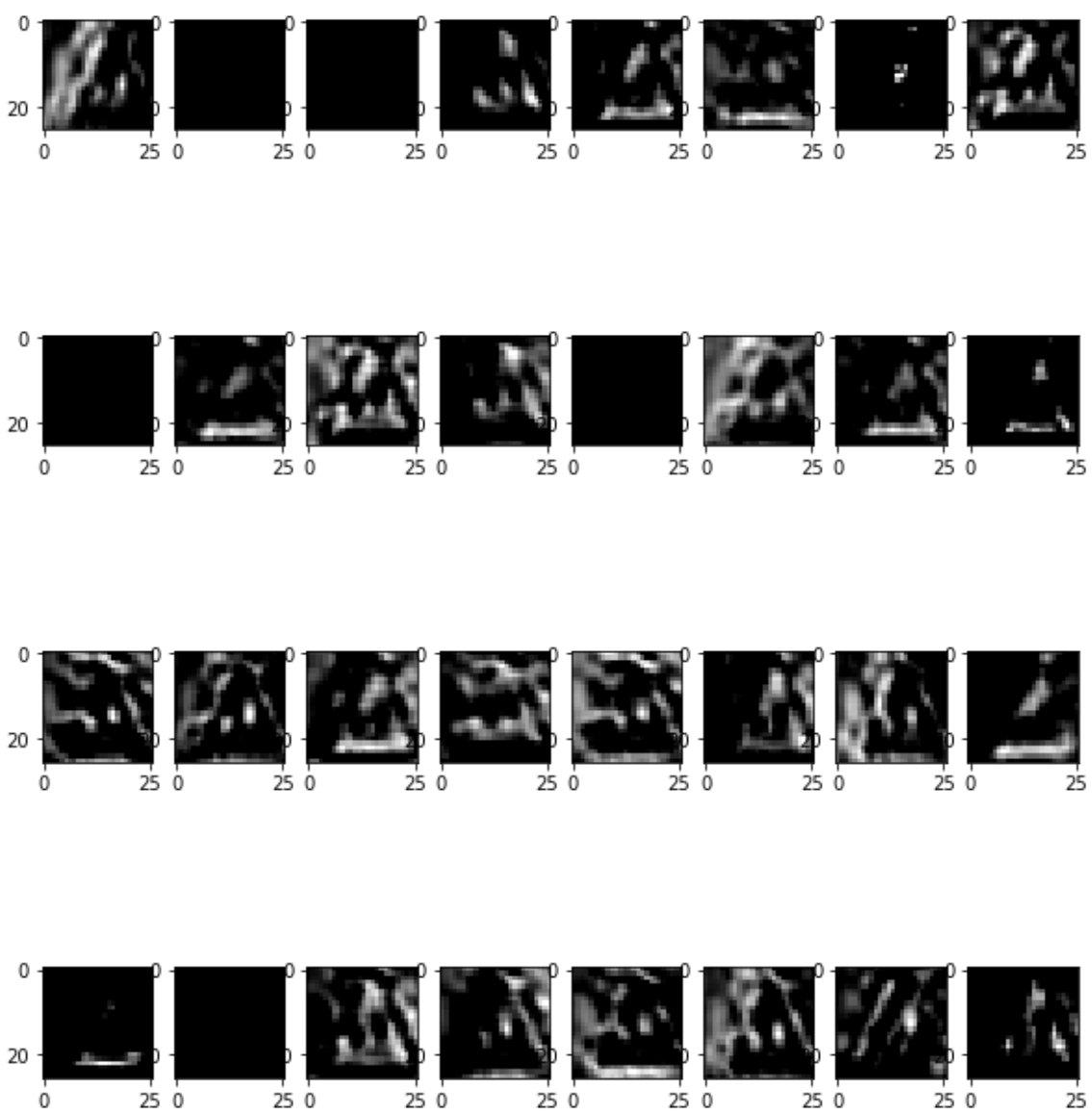


```
Probabilities of top 5 classes.
Actual class it belongs to: 28
Predicted class 23
Actual training image
```

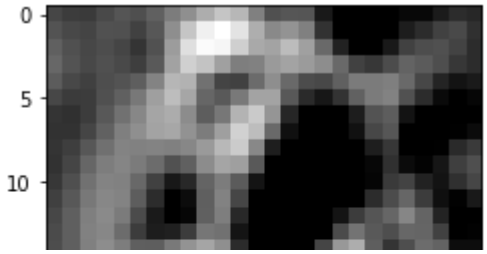



```
The kernel which activates/recognizes the shape: 13
```

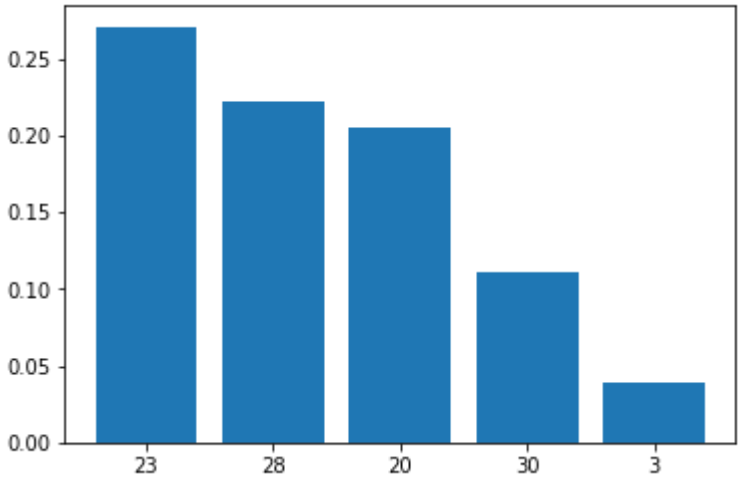




Multiple kernel activations starting from 0



Activation for 13 kernel in 3 layer.

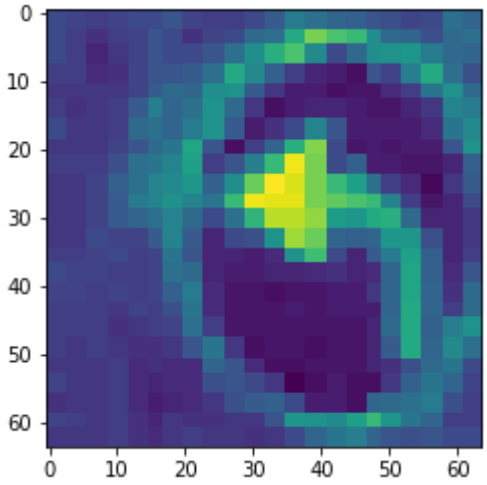


Probabilities of top 5 classes.

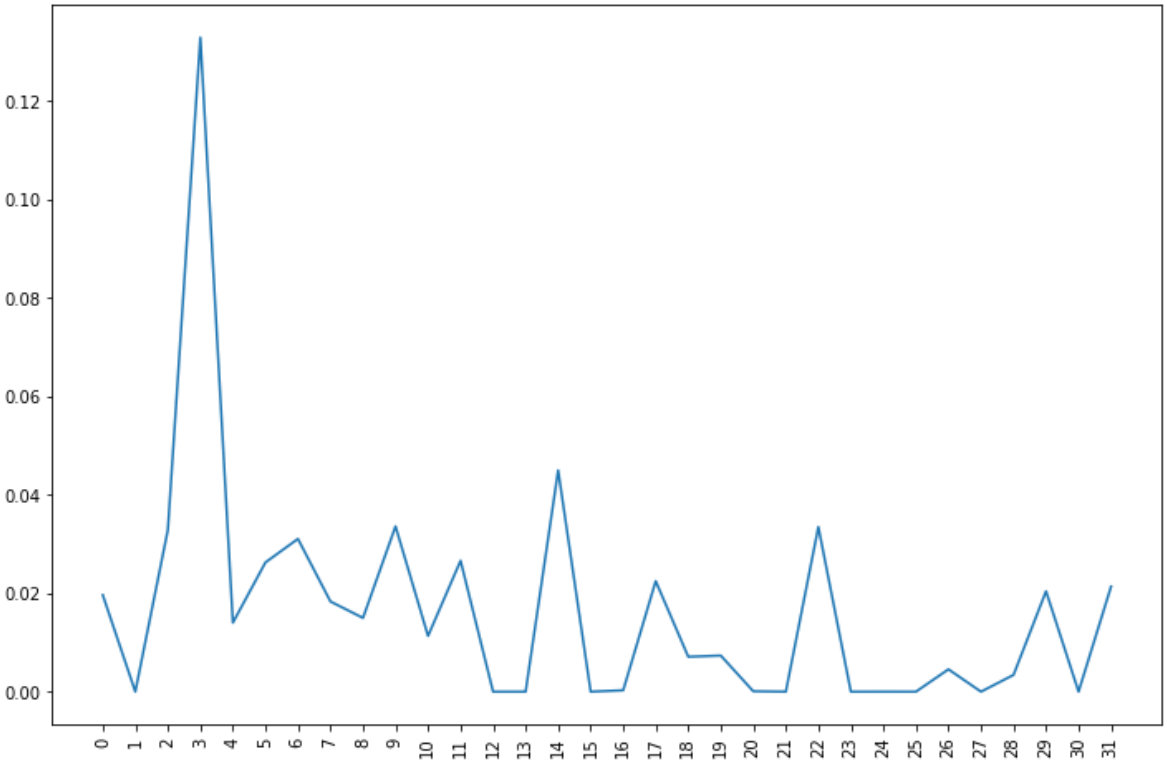
In [0]:

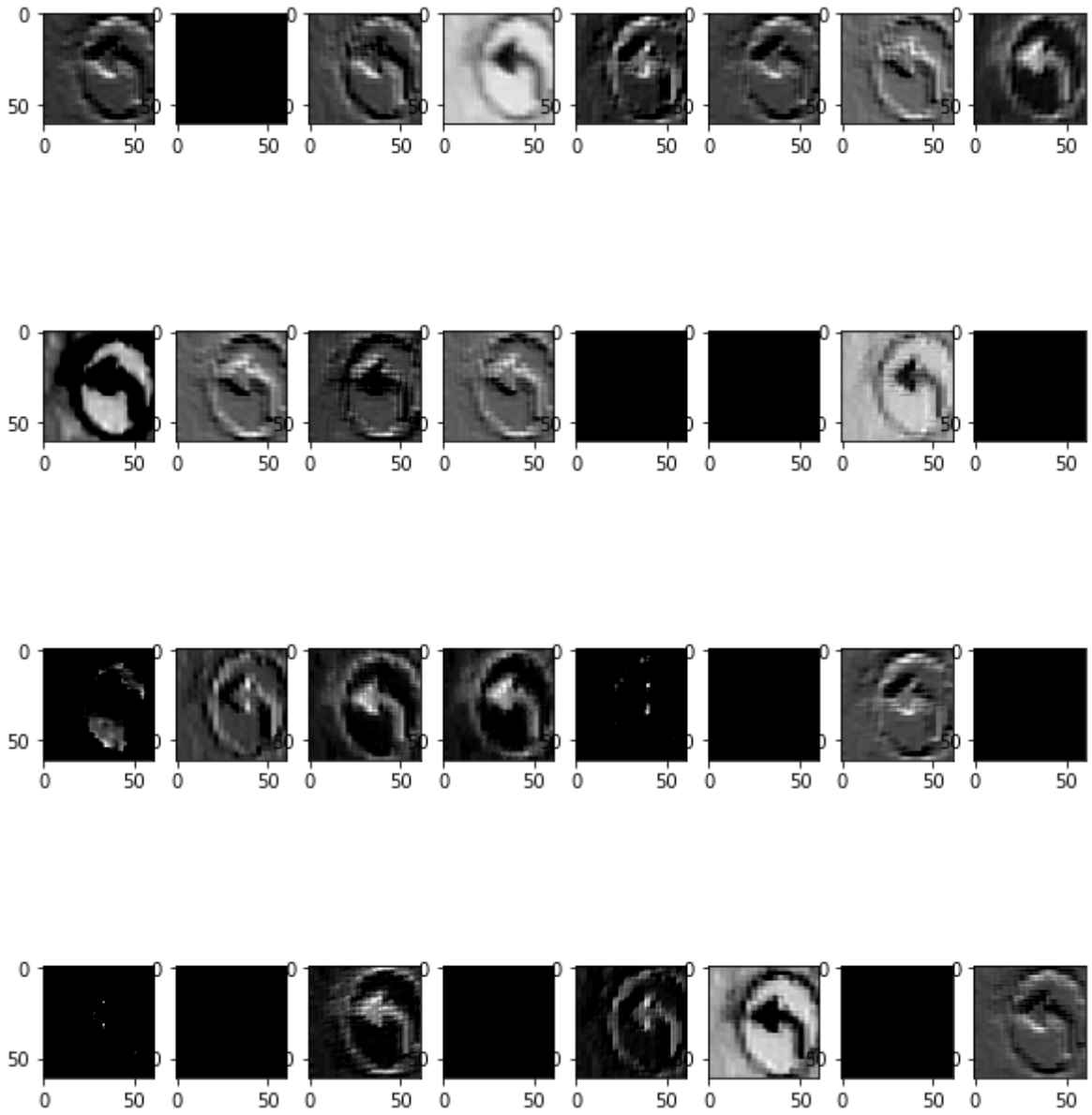
```
idx = 572
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

```
Actual class it belongs to: 34
Predicted class 9
Actual training image
```

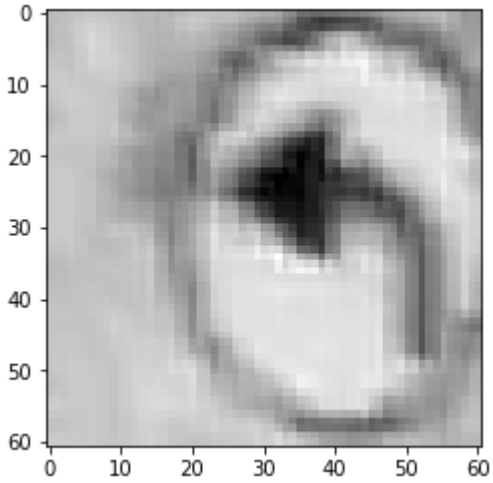



```
The kernel which activates/recognizes the shape: 3
```

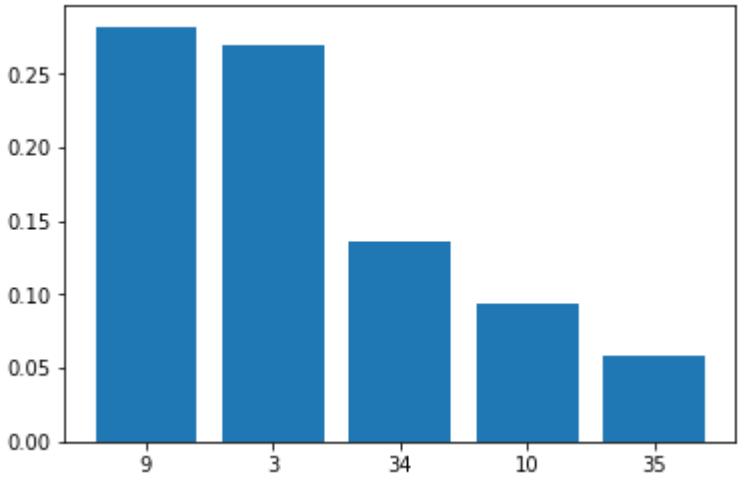




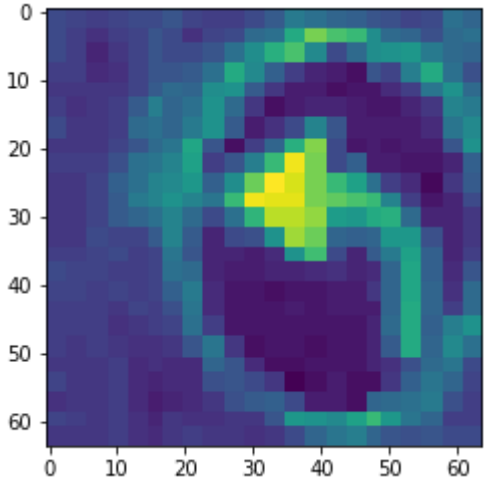
Multiple kernel activations starting from 0



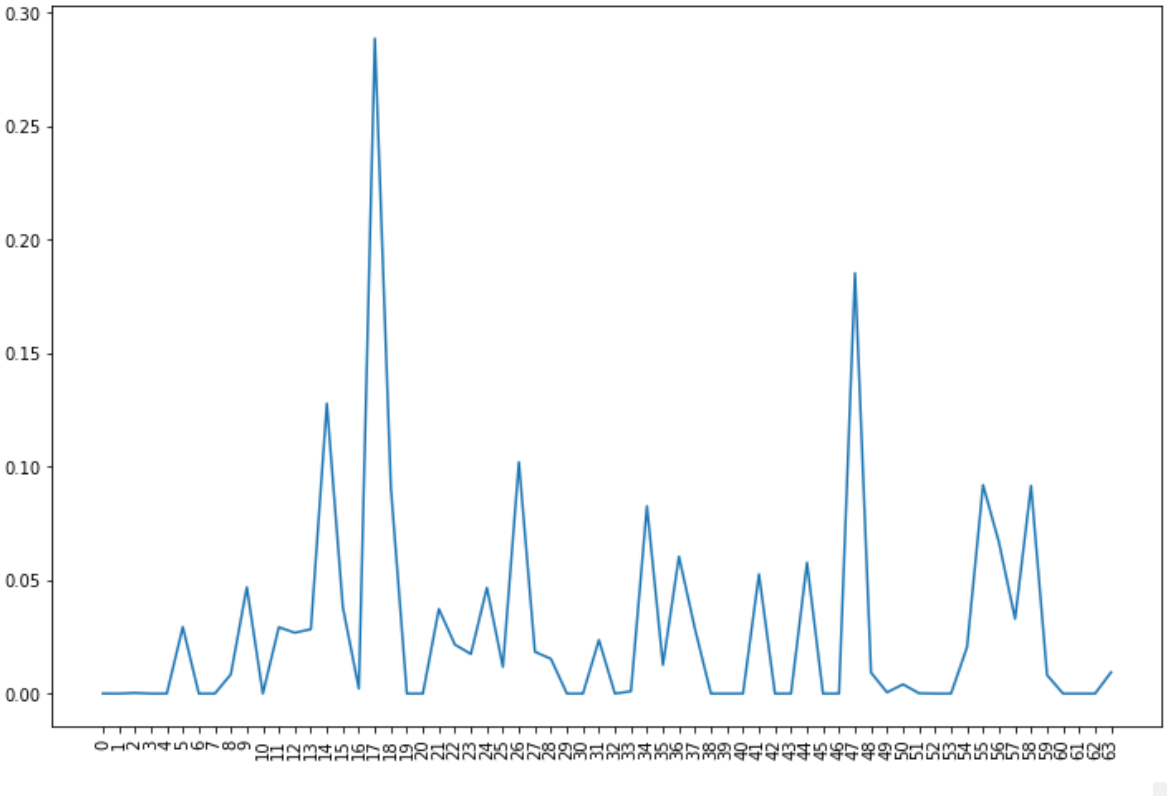
Activation for 3 kernel in 0 layer.

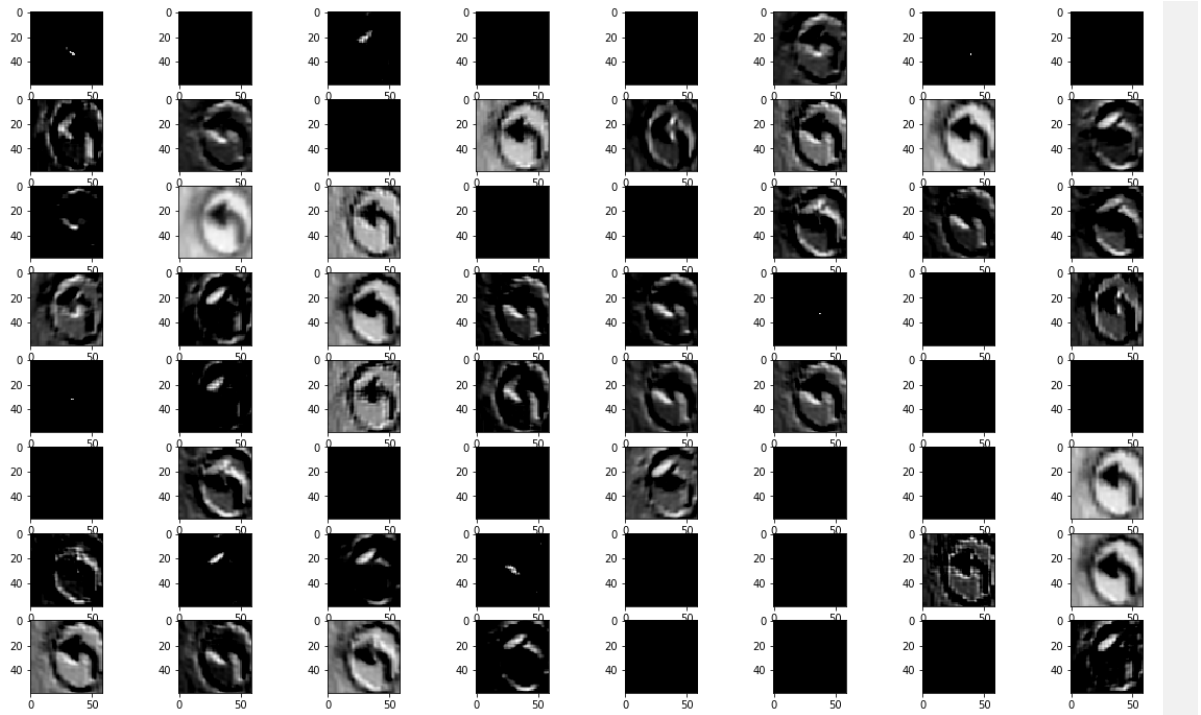



```
Probabilities of top 5 classes.
Actual class it belongs to: 34
Predicted class 9
Actual training image
```

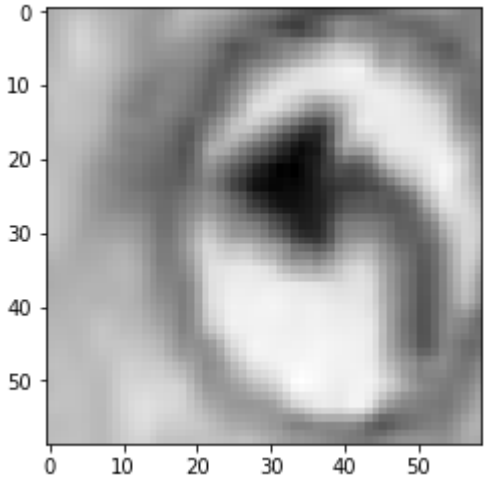



```
The kernel which activates/recognizes the shape: 17
```

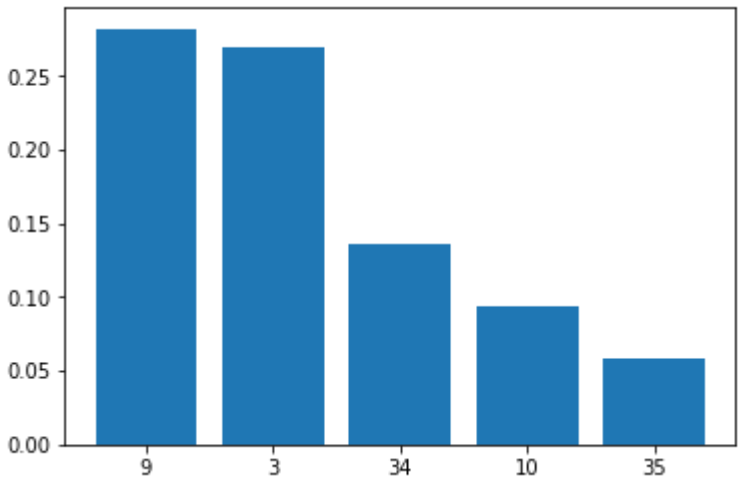




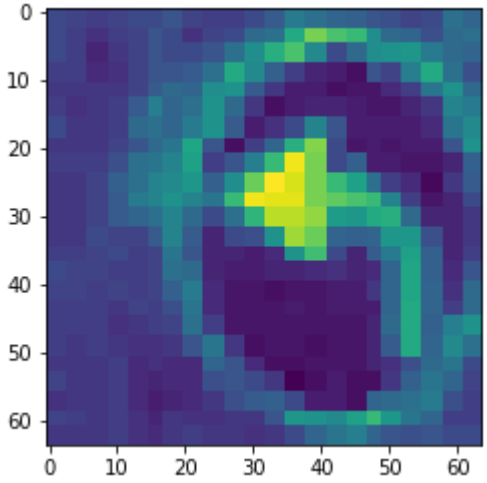
Multiple kernel activations starting from 0



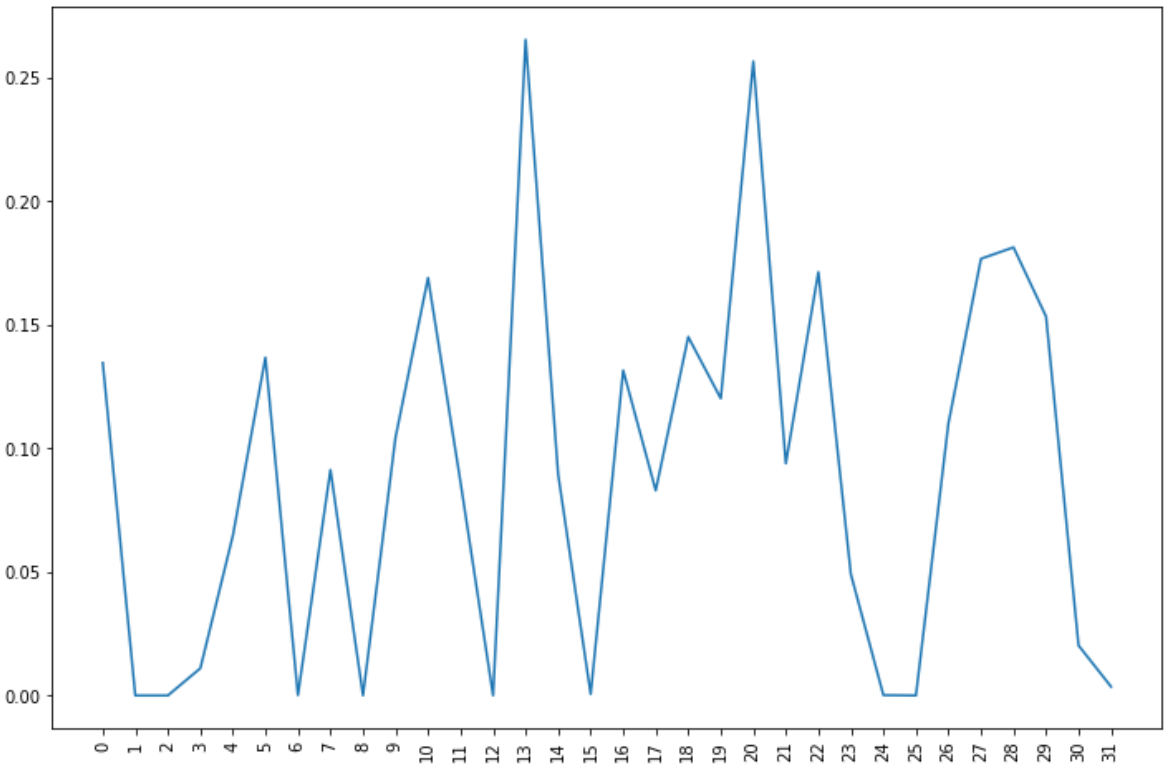
Activation for 17 kernel in 1 layer.

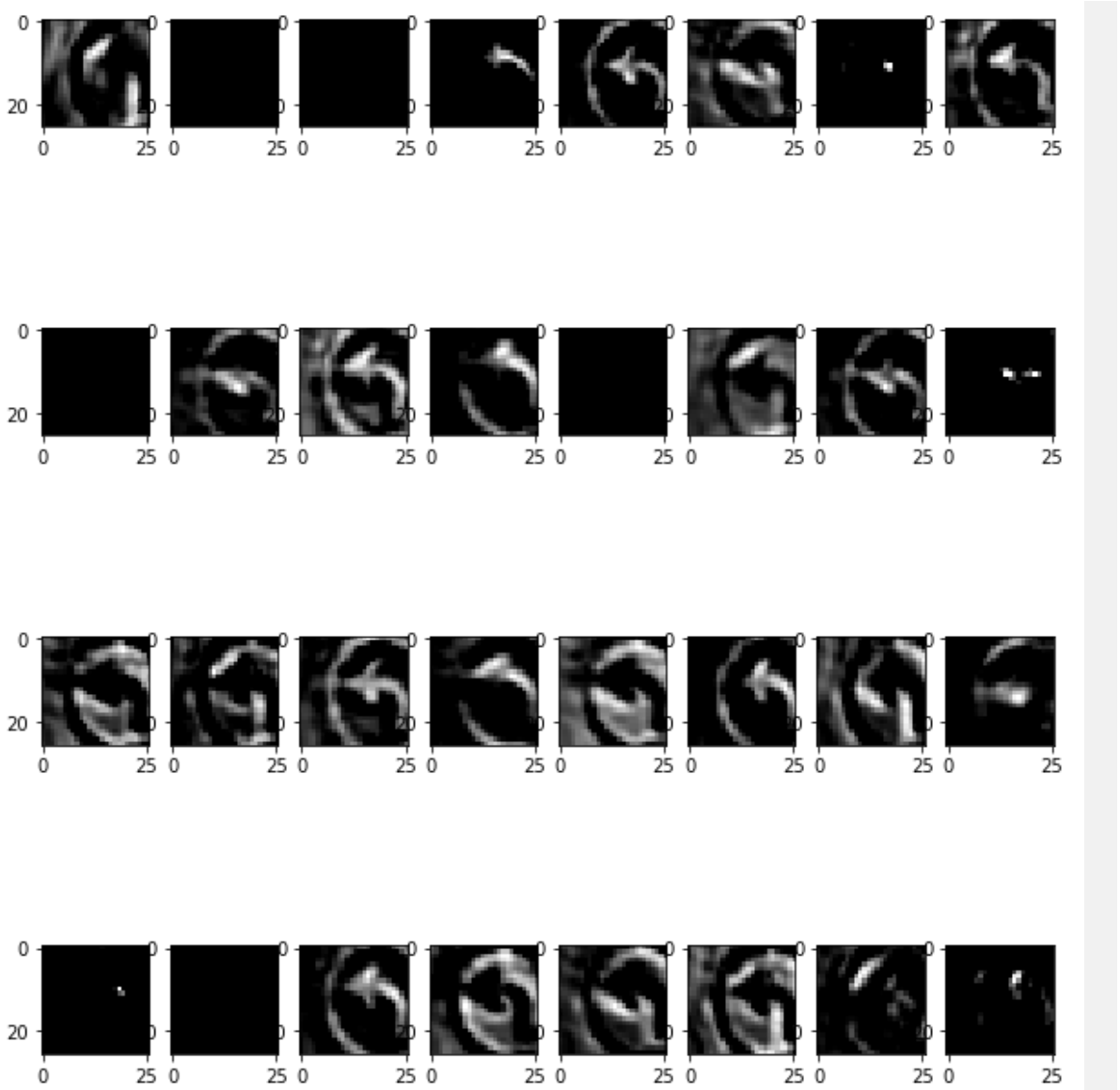


```
Probabilities of top 5 classes.
Actual class it belongs to: 34
Predicted class 9
Actual training image
```

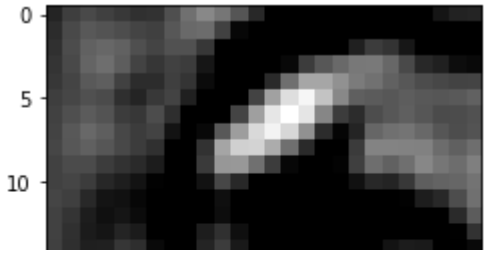



```
The kernel which activates/recognizes the shape: 13
```

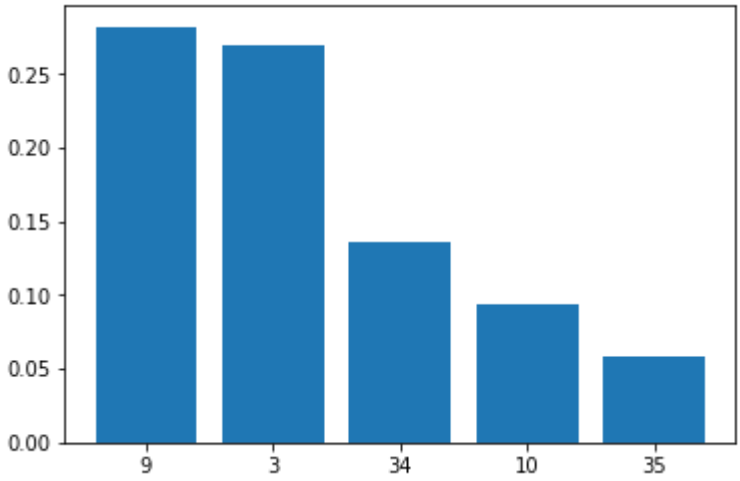




Multiple kernel activations starting from 0



Activation for 13 kernel in 3 layer.

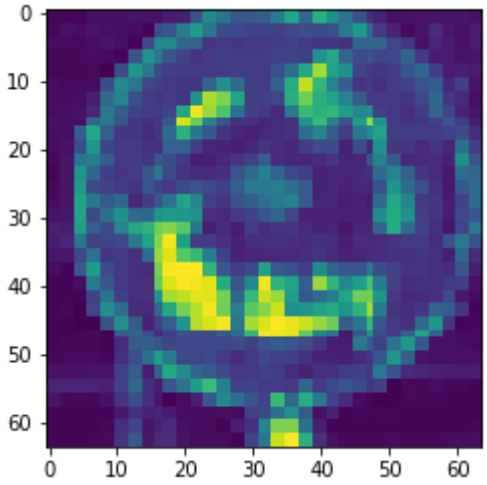


Probabilities of top 5 classes.

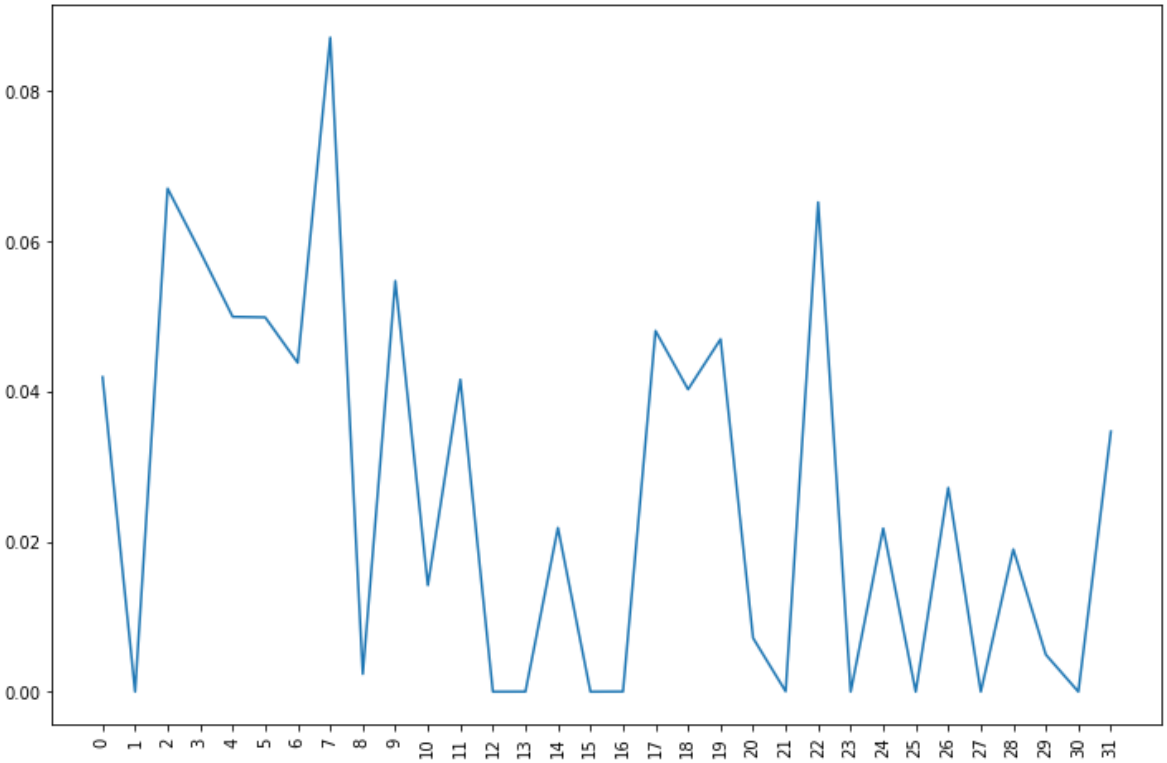
In [0]:

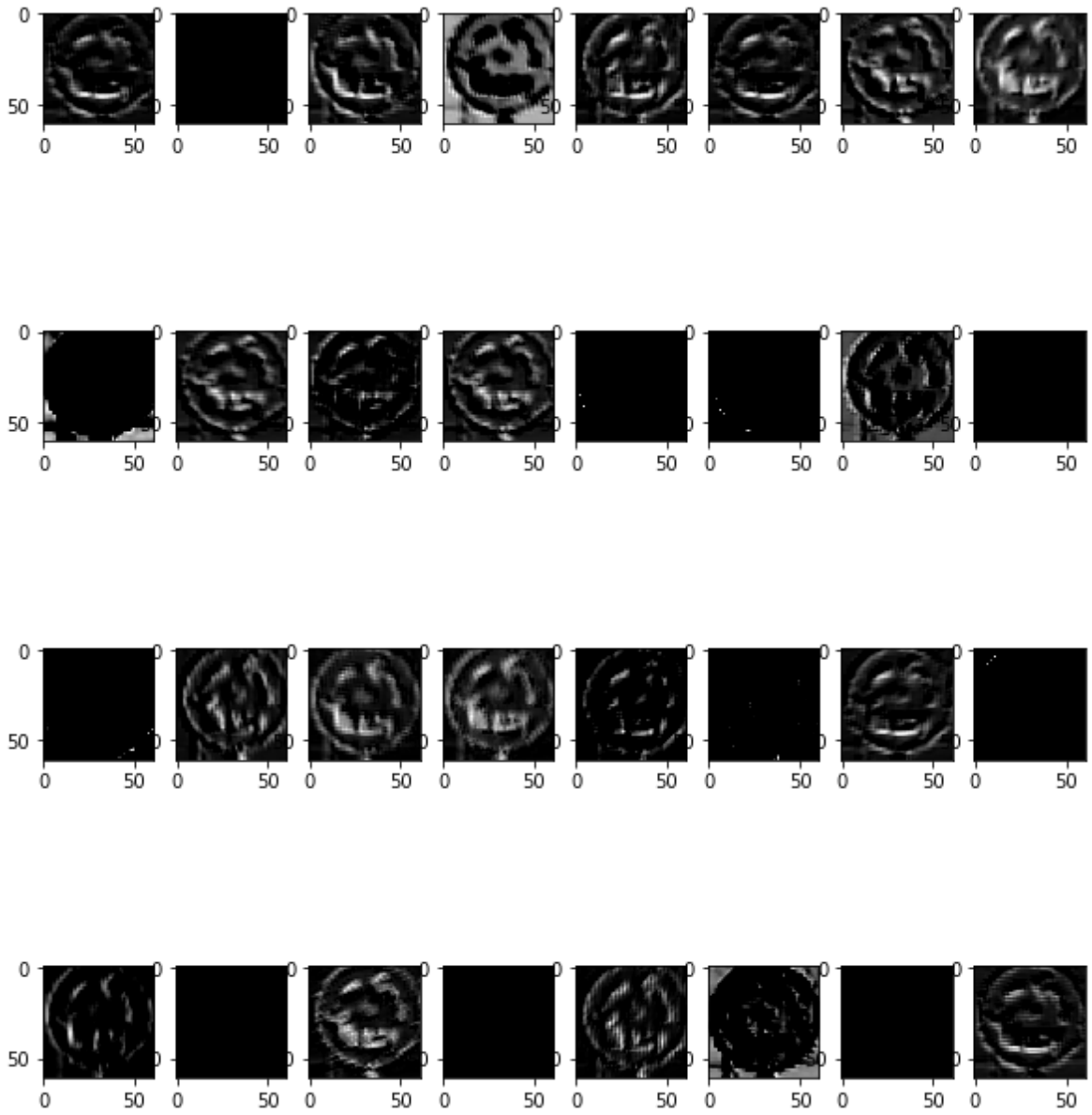
```
idx = 151
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

```
Actual class it belongs to: 40
Predicted class 2
Actual training image
```

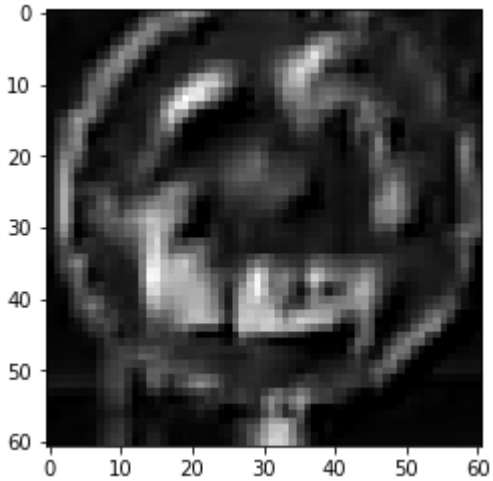



```
The kernel which activates/recognizes the shape: 7
```

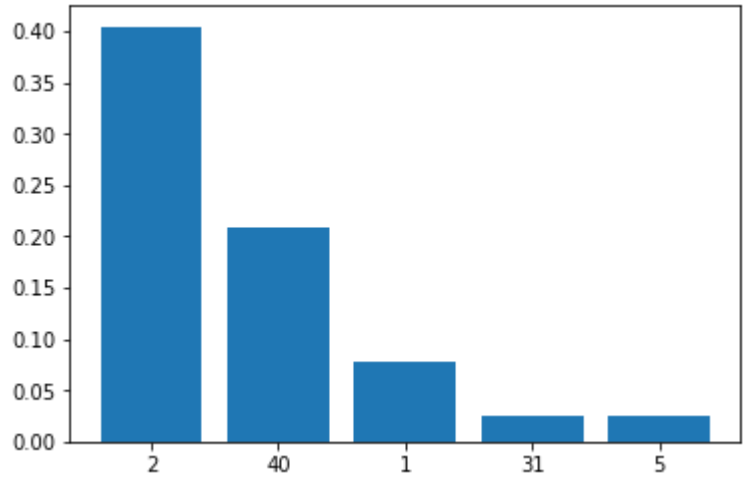




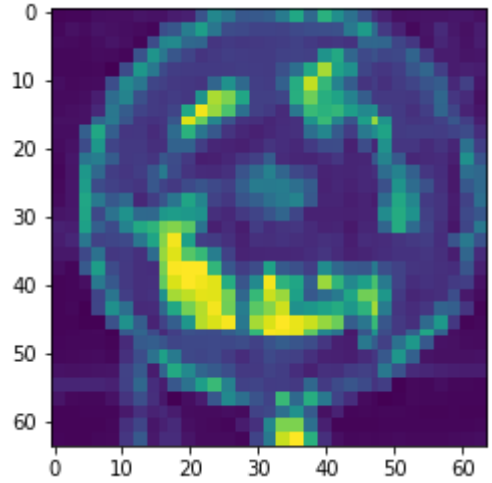
Multiple kernel activations starting from 0



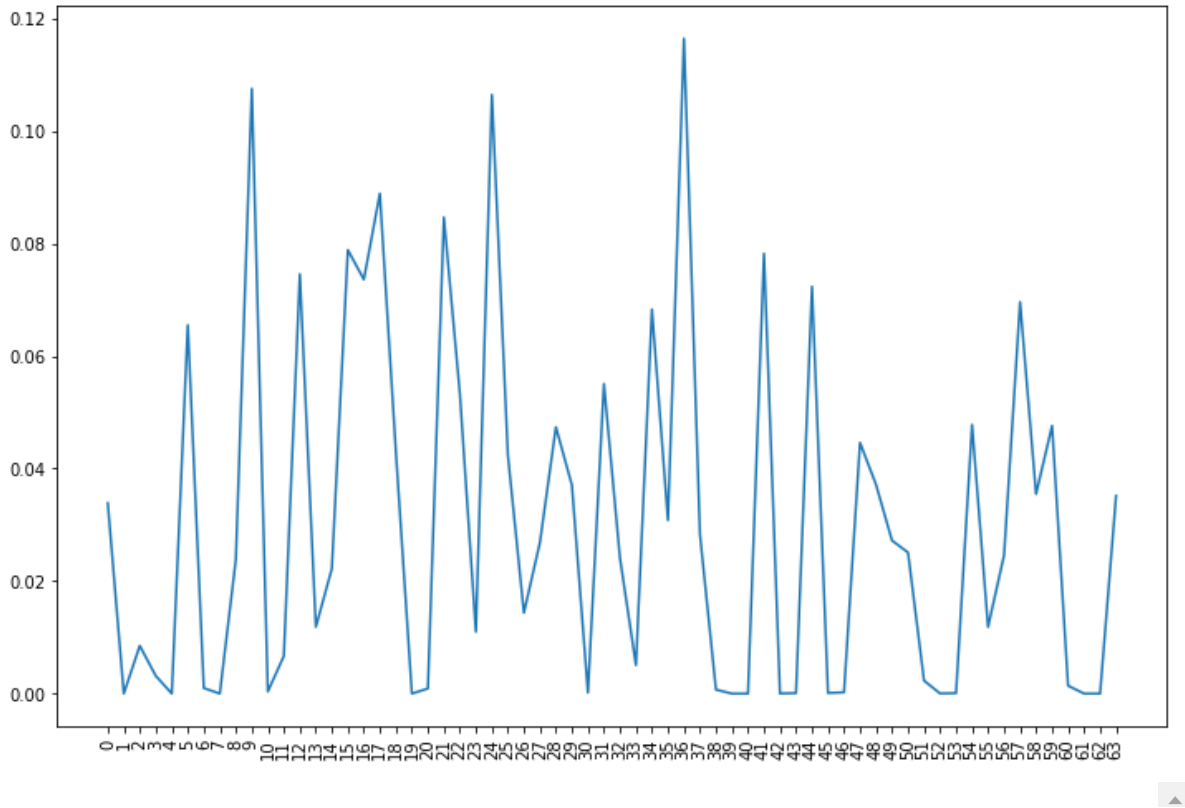
Activation for 7 kernel in 0 layer.

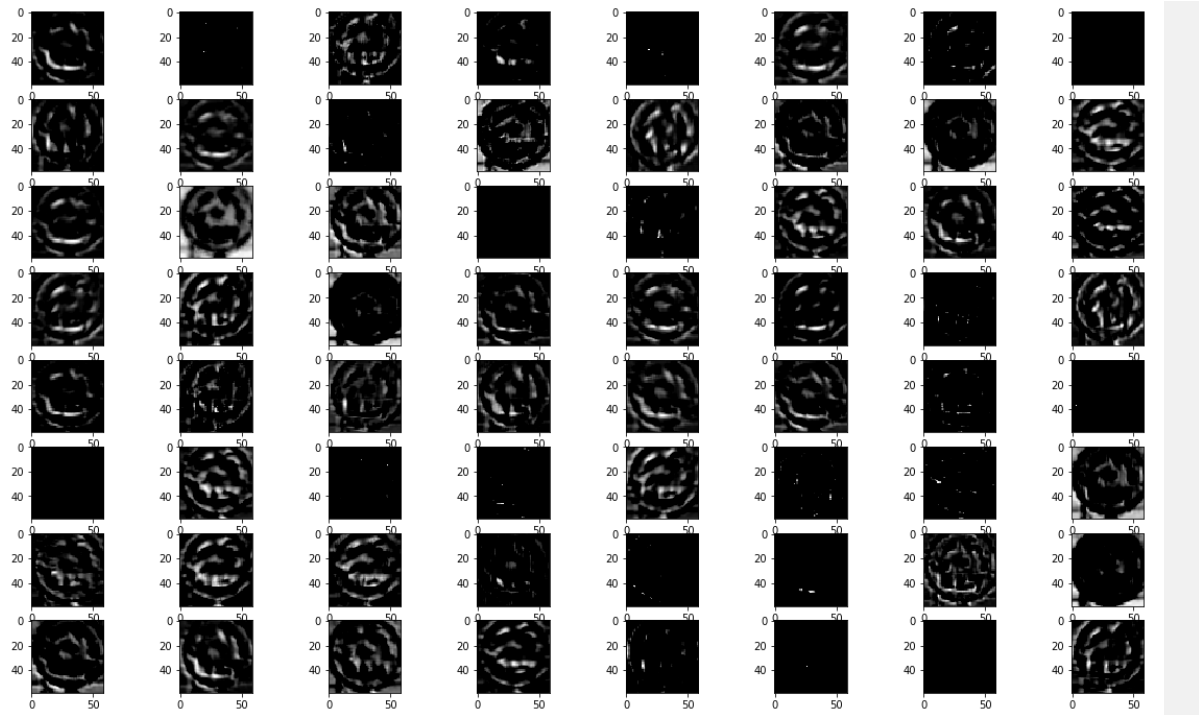



```
Probabilities of top 5 classes.
Actual class it belongs to: 40
Predicted class 2
Actual training image
```

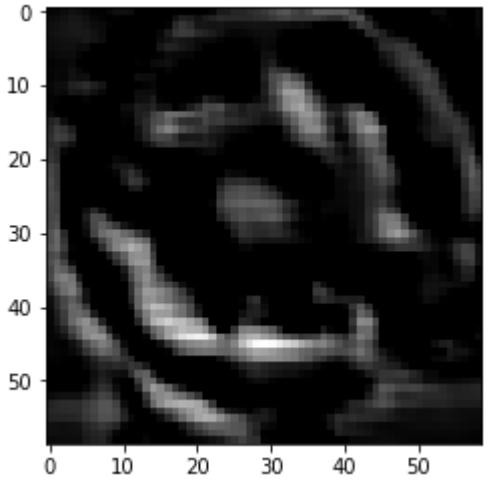



```
The kernel which activates/recognizes the shape: 36
```

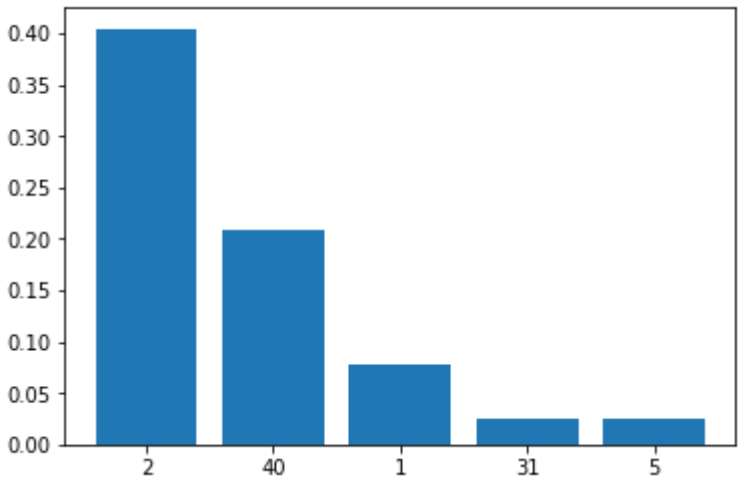




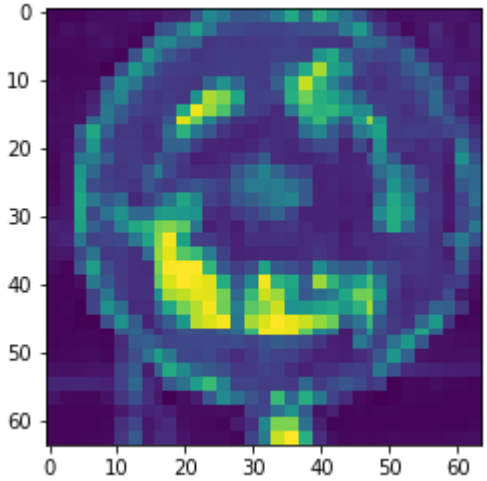
Multiple kernel activations starting from 0



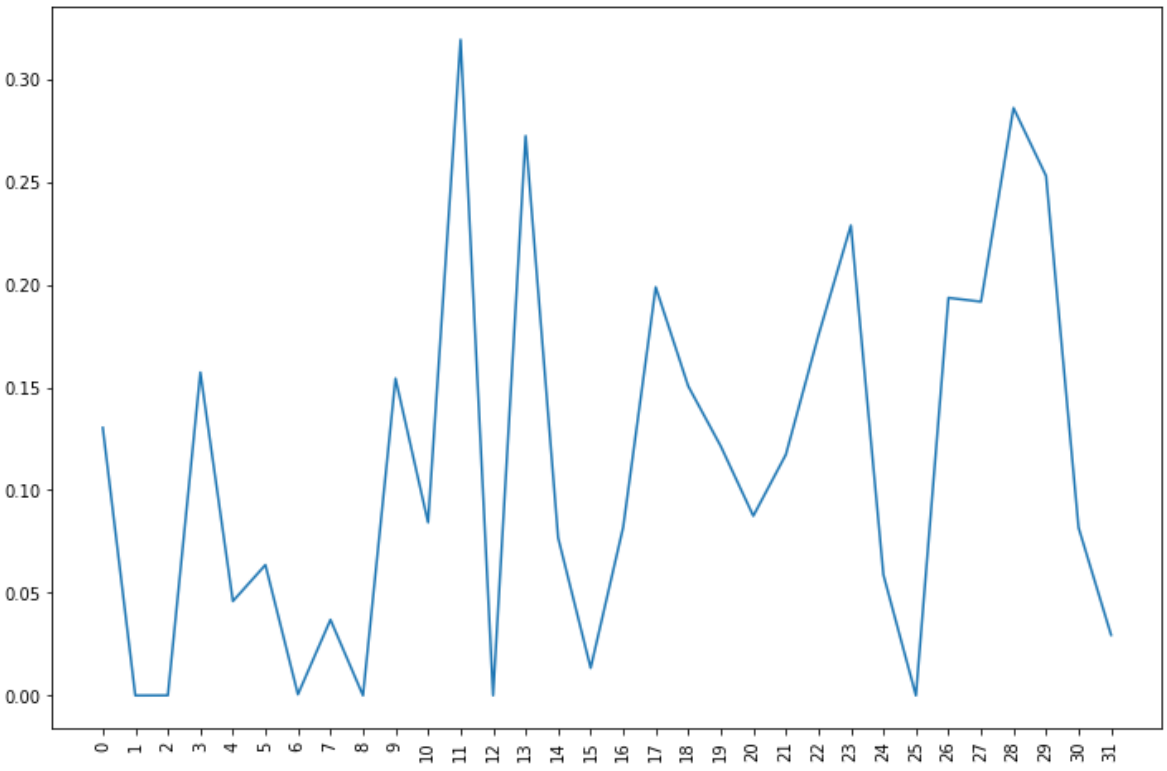
Activation for 36 kernel in 1 layer.

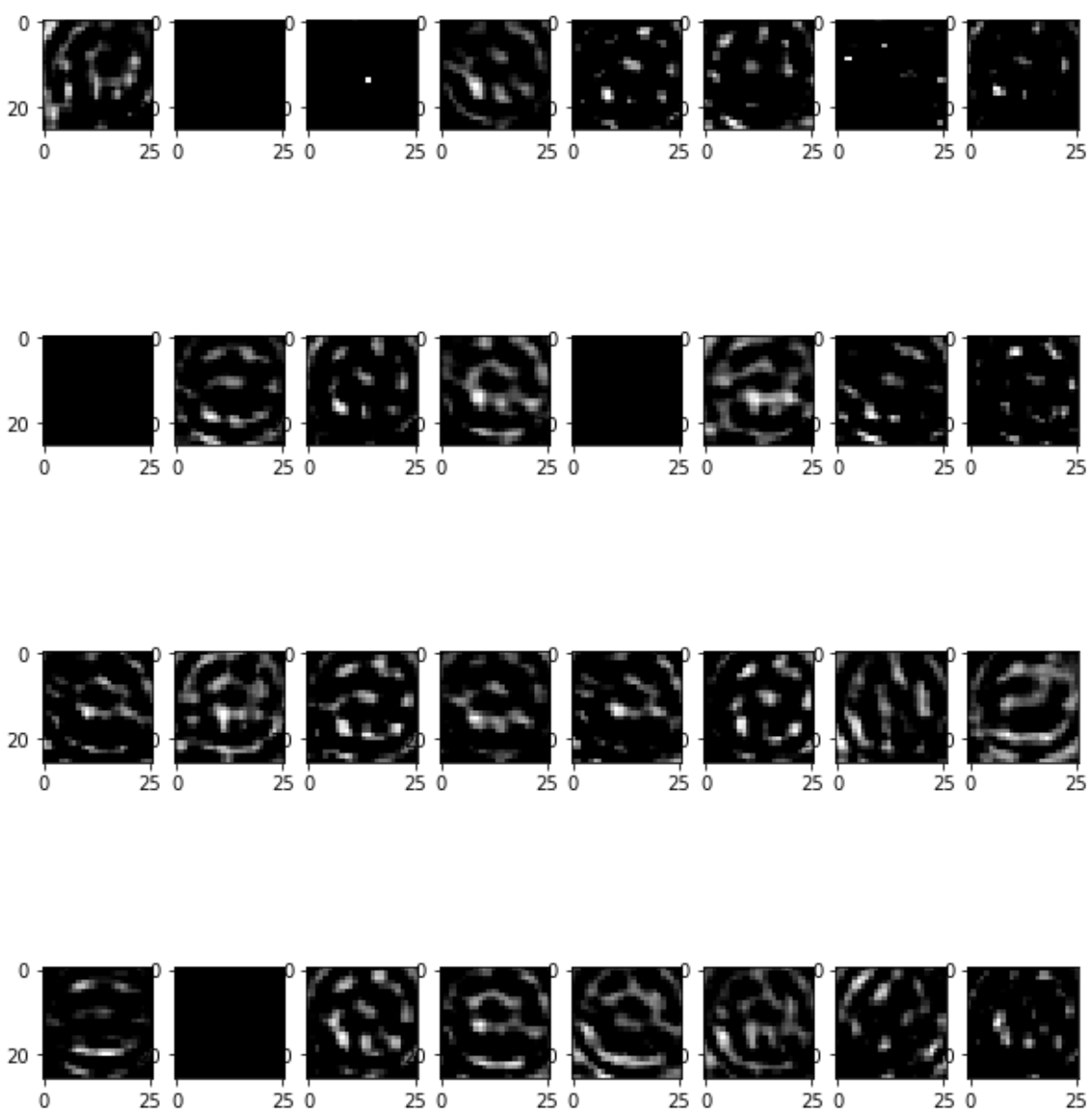


```
Probabilities of top 5 classes.
Actual class it belongs to: 40
Predicted class 2
Actual training image
```

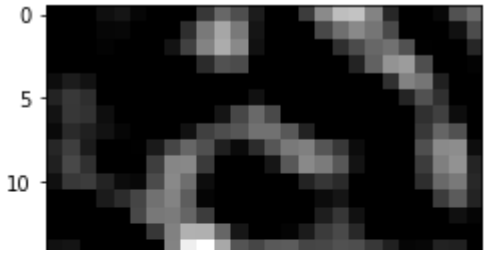



```
The kernel which activates/recognizes the shape: 11
```

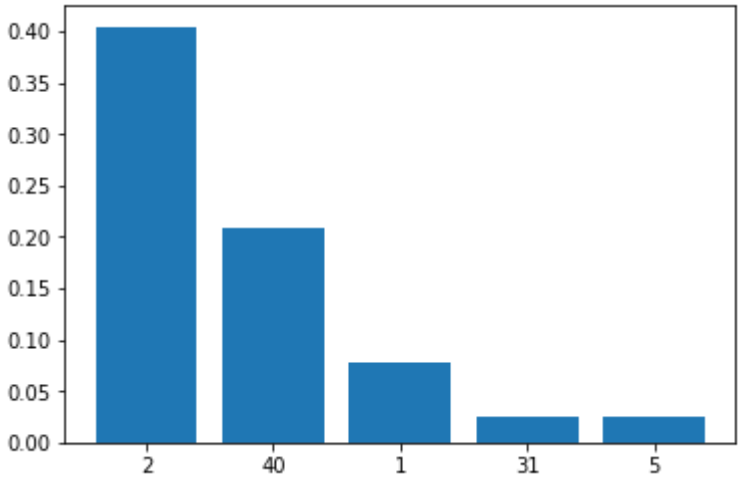




Multiple kernel activations starting from 0



Activation for 11 kernel in 3 layer.



Probabilities of top 5 classes.

[5.6] Conv(32 -- 3x3) - Conv(64 -- 3x3) - MaxPool(2x2) - Conv(32 -- 3x3) - Dropout(0.75) - MaxPool(2x2) - Conv(32 -- 3x3) - Dense(256) - Dropout(0.5) - Dense(128) - Dropout(0.5) with class weights

In [0]:

```
from sklearn.utils import class_weight
class_weights = class_weight.compute_class_weight(
    'balanced', np.unique(np.argmax(y_test, axis=1)),
    np.argmax(y_test, axis=1)
)
class_weights = dict(enumerate(class_weights))
class_weights
```

Out[115]:

```
{0: 4.342192691029901,
 1: 0.426767768385154,
 2: 0.404671804045686,
 3: 0.6332364341085271,
 4: 0.46523493118177506,
 5: 0.45517161320280153,
 6: 2.154001098699872,
 7: 0.6106208471760798,
 8: 0.652883387911417,
 9: 0.6065590677048419,
 10: 0.46131220832189496,
 11: 0.7032342918634543,
 12: 0.42944762878317694,
 13: 0.42085867620751344,
 14: 1.2723634396971335,
 15: 1.3151833631484795,
 16: 2.154001098699872,
 17: 0.7952271498107085,
 18: 0.8264596360570505,
 19: 4.799265605875153,
 20: 2.5097077021548966,
 21: 3.218331053351573,
 22: 2.486892177589852,
 23: 1.7097383720930233,
 24: 3.218331053351573,
 25: 0.5972885142683051,
 26: 1.4867290192113245,
 27: 3.7994186046511627,
 28: 1.6991188790986567,
 29: 4.022913816689466,
 30: 1.823720930232558,
 31: 1.0898730658760307,
 32: 3.462761259935237,
 33: 1.3816067653276956,
 34: 2.260811070536229,
 35: 0.8117452211717617,
 36: 2.5807371654234315,
 37: 4.884966777408638,
 38: 0.4448099829835508,
 39: 2.630366726296959,
 40: 2.2422798322531454,
 41: 3.6967316153362666,
 42: 4.484559664506291}
```

In [0]:

```
epochs = 20
model = Sequential()
model.add(Conv2D(
    32, kernel_size=(3, 3), activation='relu', input_shape=input_shape
))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(Dropout(rate=0.75))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(
    loss=keras.losses.categorical_crossentropy,
    optimizer=keras.optimizers.Adadelta(), metrics=['accuracy']
)

model.summary()

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 62, 62, 32)        320
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 60, 60, 64)        18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 28, 28, 32)        18464
_________________________________________________________________
dropout_20 (Dropout)         (None, 28, 28, 32)        0
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 12, 12, 32)        9248
_________________________________________________________________
flatten_9 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_20 (Dense)             (None, 256)               1179904
_________________________________________________________________
dropout_21 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_21 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_22 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_22 (Dense)             (None, 43)                5547
```

```
=================================================================
Total params: 1,264,875
Trainable params: 1,264,875
Non-trainable params: 0
_________________________________________________________________
Train on 27446 samples, validate on 11763 samples
Epoch 1/20
27446/27446 [==============================] - 10s 365us/step - loss: 3.1364
- acc: 0.1798 - val_loss: 1.9691 - val_acc: 0.6734
Epoch 2/20
27446/27446 [==============================] - 9s 326us/step - loss: 1.1106
- acc: 0.6557 - val_loss: 0.7509 - val_acc: 0.9077
Epoch 3/20
27446/27446 [==============================] - 9s 328us/step - loss: 0.5846
- acc: 0.8111 - val_loss: 0.5091 - val_acc: 0.9526
Epoch 4/20
27446/27446 [==============================] - 9s 329us/step - loss: 0.4027
- acc: 0.8669 - val_loss: 0.6244 - val_acc: 0.9360
Epoch 5/20
27446/27446 [==============================] - 9s 332us/step - loss: 0.3114
- acc: 0.8970 - val_loss: 0.2269 - val_acc: 0.9711
Epoch 6/20
27446/27446 [==============================] - 9s 333us/step - loss: 0.2520
- acc: 0.9194 - val_loss: 0.2160 - val_acc: 0.9736
Epoch 7/20
27446/27446 [==============================] - 9s 331us/step - loss: 0.2113
- acc: 0.9311 - val_loss: 0.1674 - val_acc: 0.9754
Epoch 8/20
27446/27446 [==============================] - 9s 329us/step - loss: 0.1900
- acc: 0.9396 - val_loss: 0.1460 - val_acc: 0.9765
Epoch 9/20
27446/27446 [==============================] - 9s 328us/step - loss: 0.1627
- acc: 0.9474 - val_loss: 0.2898 - val_acc: 0.9641
Epoch 10/20
27446/27446 [==============================] - 9s 327us/step - loss: 0.1419
- acc: 0.9525 - val_loss: 0.1342 - val_acc: 0.9814
Epoch 11/20
27446/27446 [==============================] - 9s 326us/step - loss: 0.1378
- acc: 0.9547 - val_loss: 0.0948 - val_acc: 0.9814
Epoch 12/20
27446/27446 [==============================] - 9s 327us/step - loss: 0.1289
- acc: 0.9593 - val_loss: 0.1161 - val_acc: 0.9852
Epoch 13/20
27446/27446 [==============================] - 9s 326us/step - loss: 0.1158
- acc: 0.9634 - val_loss: 0.1038 - val_acc: 0.9842
Epoch 14/20
27446/27446 [==============================] - 9s 328us/step - loss: 0.1032
- acc: 0.9667 - val_loss: 0.0861 - val_acc: 0.9851
Epoch 15/20
27446/27446 [==============================] - 9s 328us/step - loss: 0.0960
- acc: 0.9690 - val_loss: 0.0789 - val_acc: 0.9874
Epoch 16/20
27446/27446 [==============================] - 9s 329us/step - loss: 0.0956
- acc: 0.9700 - val_loss: 0.0889 - val_acc: 0.9879
Epoch 17/20
27446/27446 [==============================] - 9s 327us/step - loss: 0.0799
- acc: 0.9734 - val_loss: 0.0692 - val_acc: 0.9889
Epoch 18/20
27446/27446 [==============================] - 9s 327us/step - loss: 0.0760
- acc: 0.9753 - val_loss: 0.0619 - val_acc: 0.9884
Epoch 19/20
```

```
27446/27446 [==============================] - 9s 327us/step - loss: 0.0742
- acc: 0.9764 - val_loss: 0.0582 - val_acc: 0.9884
Epoch 20/20
27446/27446 [==============================] - 9s 327us/step - loss: 0.0691
- acc: 0.9776 - val_loss: 0.0766 - val_acc: 0.9877
```

In [0]:

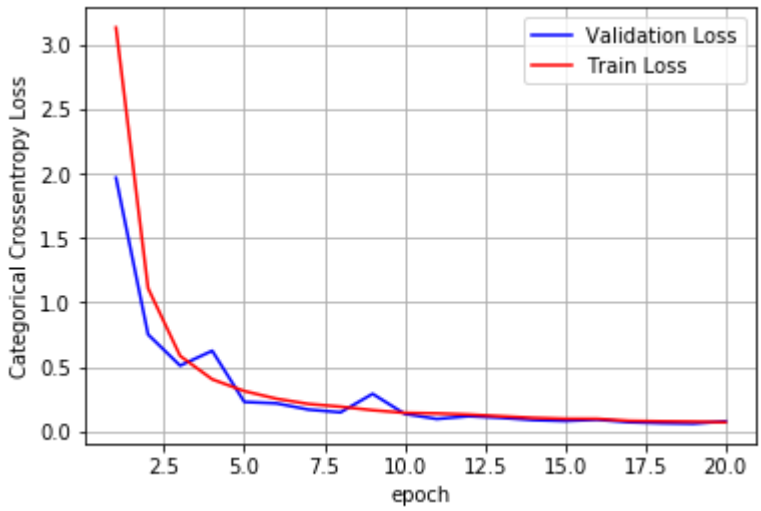
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(fig, x, vy, ty, ax)
```

```
Test score: 0.07662140248161407
Test accuracy: 0.9876732126158293
```



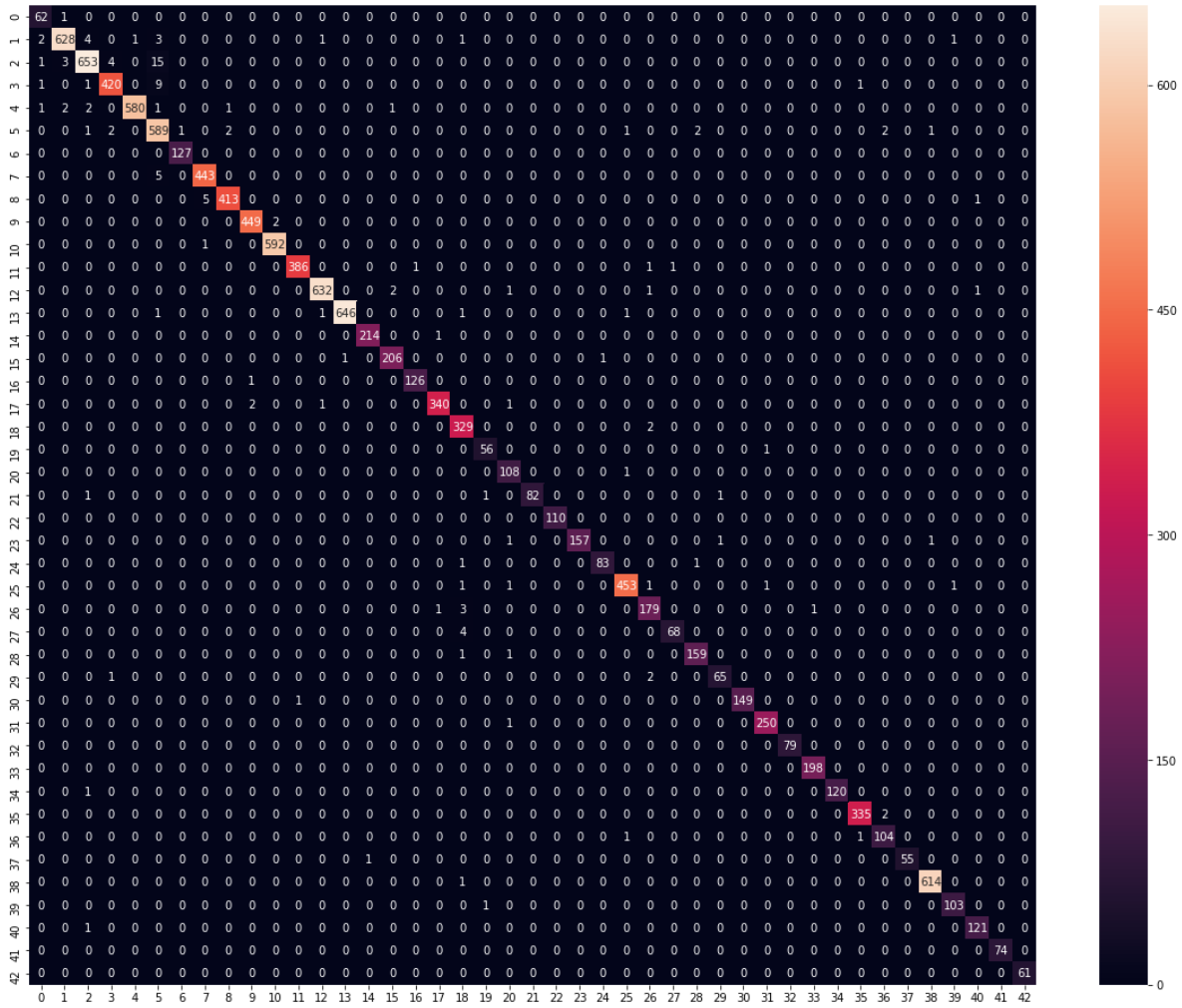
In [0]:

```
plt.figure(figsize=(20,16))
y_prediction = model.predict(x_test)
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_prediction, axis = 1)
# Convert validation observations to one hot vectors
y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
print("-----------------------------------------------------------")
print("Micro F1 score", f1_score(y_true, y_pred_classes, average='micro'))
print("-----------------------------------------------------------")
confusion_mtx = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(confusion_mtx, annot=True, fmt="d")
```

```
-----------------------------------------------------------
Micro F1 score 0.9876732126158293
-----------------------------------------------------------
```

Out[127]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8afc582cf8>
```



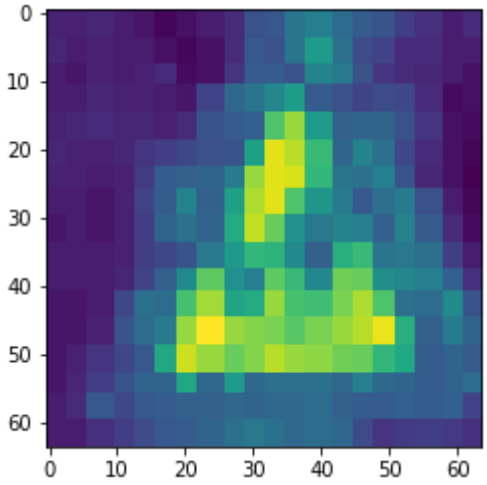
In [0]:

```
layer_outputs = [layer.output for layer in model.layers]
activation_model = Model(inputs=model.input, outputs=layer_outputs)
```

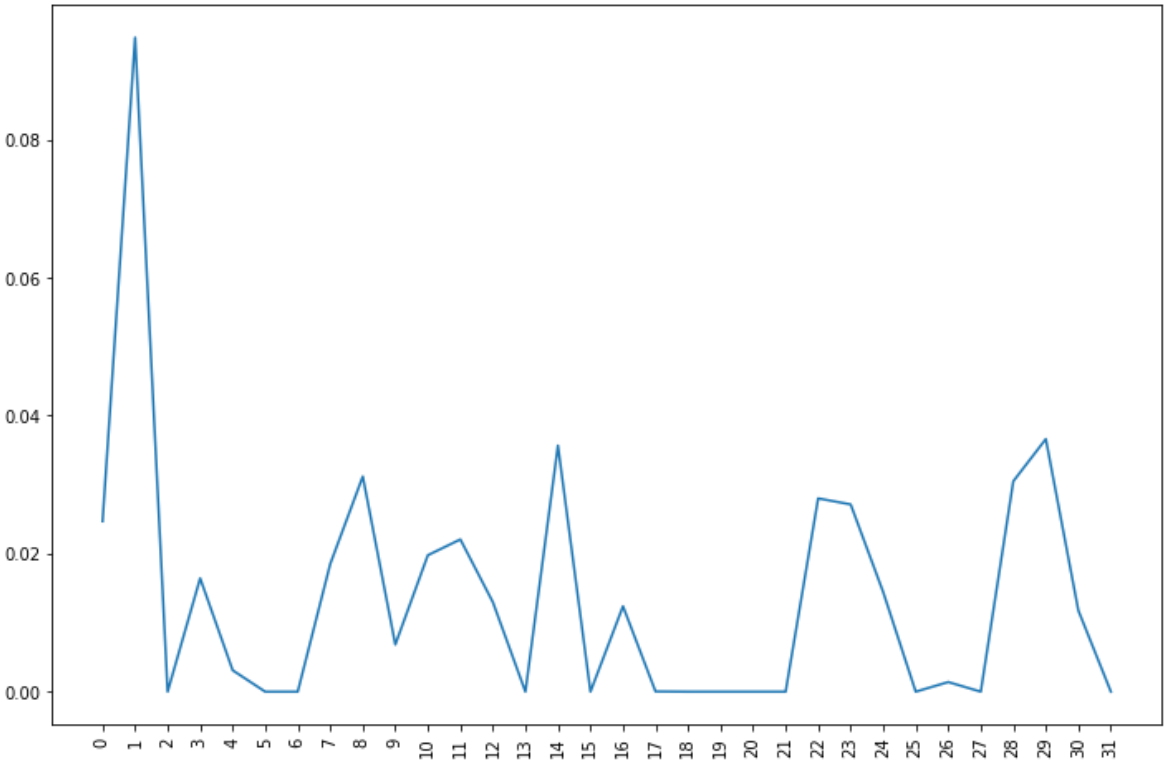
In [0]:

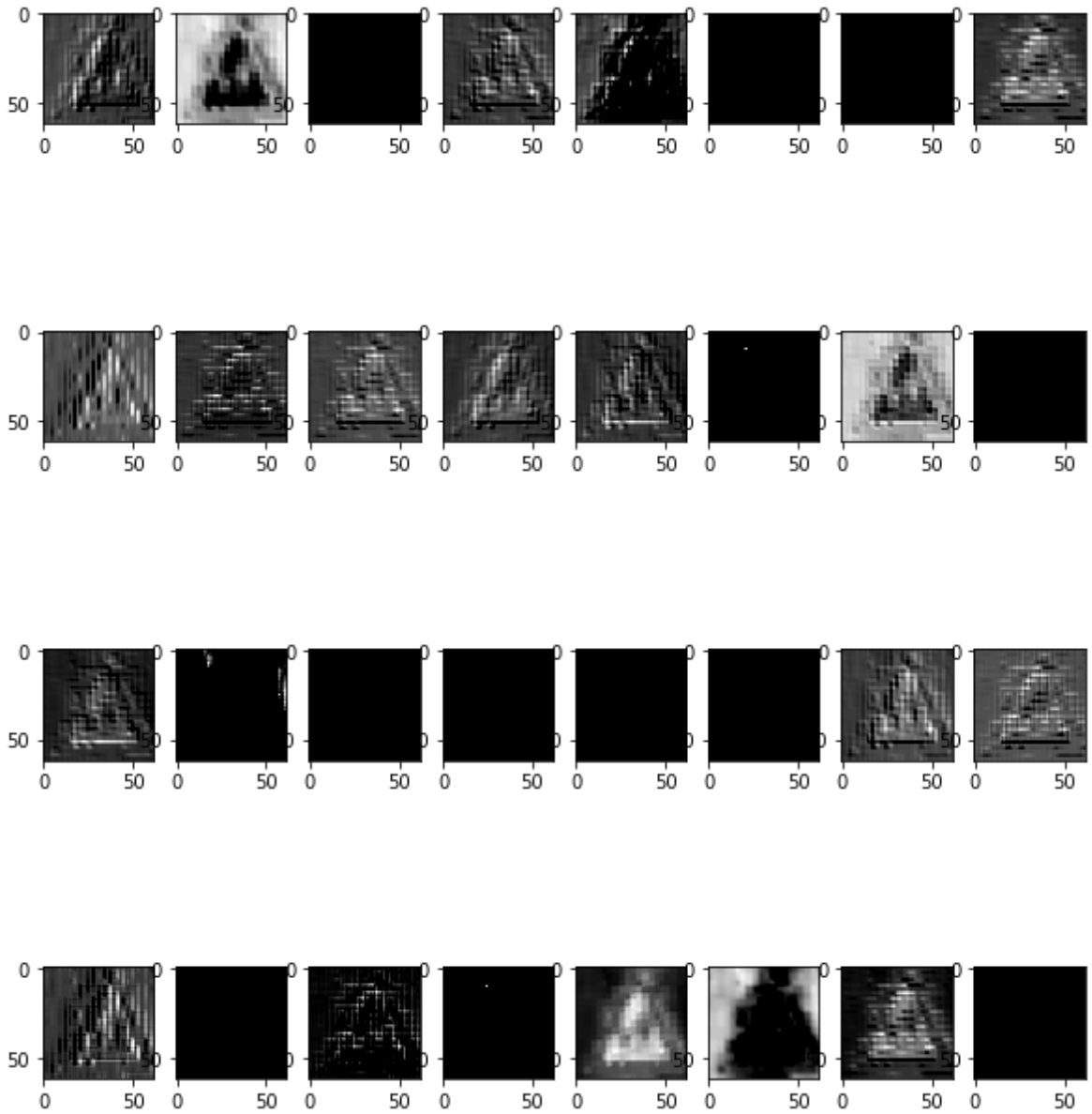
```
idx = 7210
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

```
Actual class it belongs to: 28
Predicted class 28
Actual training image
```

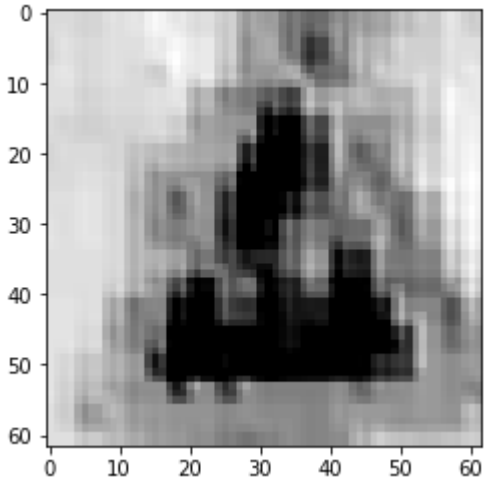



```
The kernel which activates/recognizes the shape: 1
```

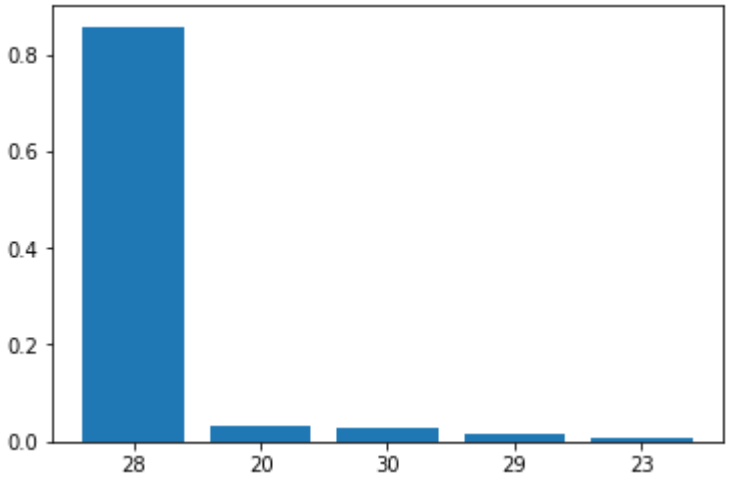




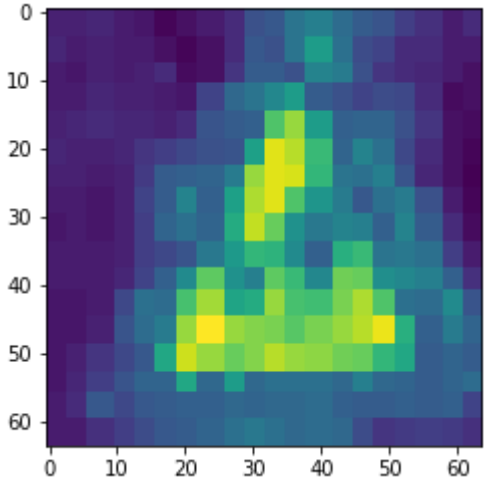
Multiple kernel activations starting from 0



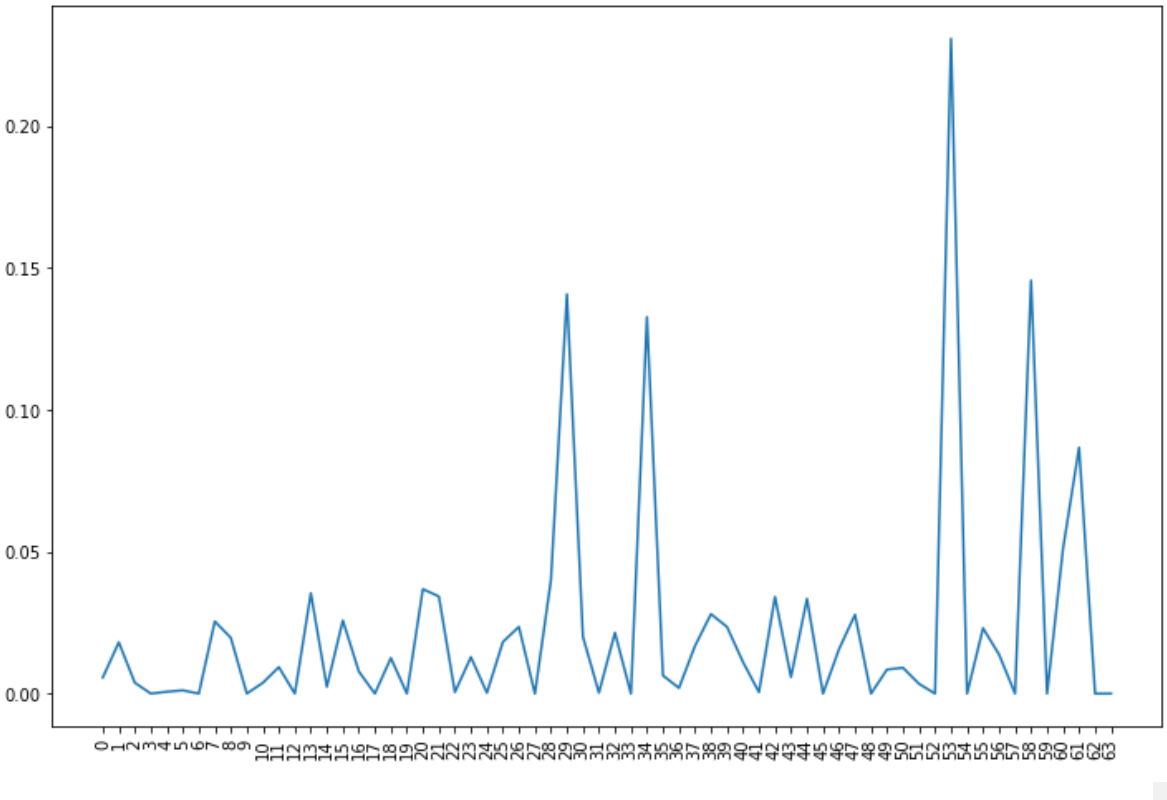
Activation for 1 kernel in 0 layer.



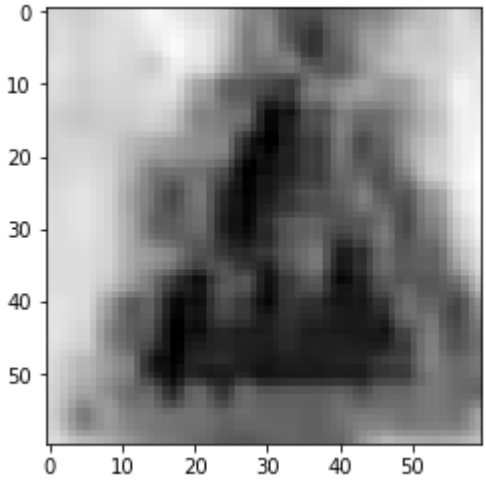

```
Probabilities of top 5 classes.
Actual class it belongs to: 28
Predicted class 28
Actual training image
```



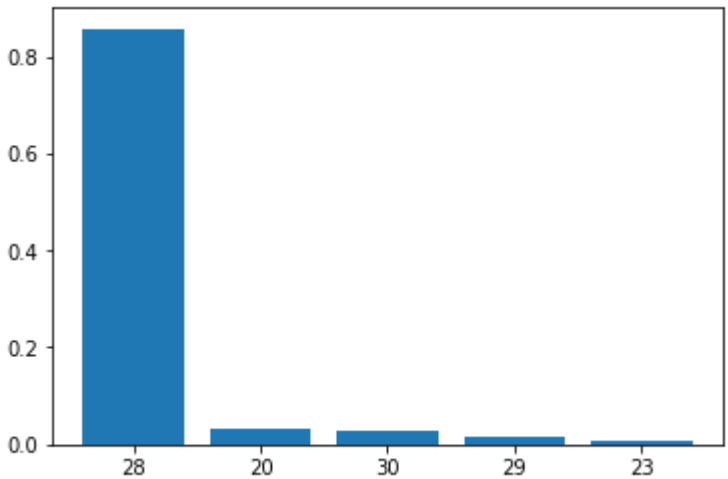

```
The kernel which activates/recognizes the shape: 53
```



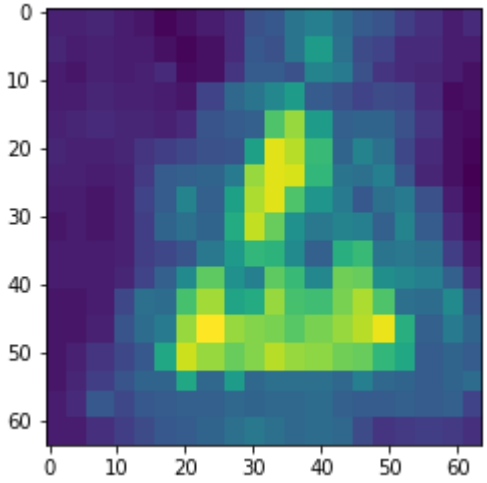
Multiple kernel activations starting from 0



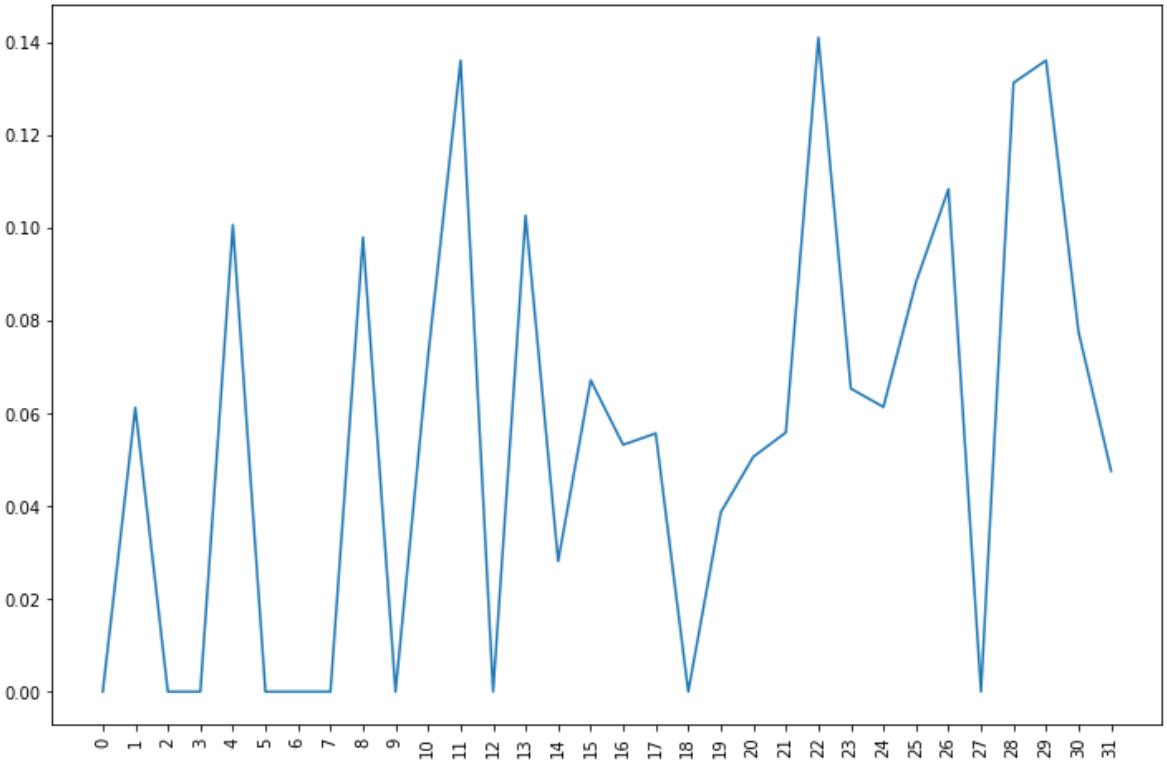
Activation for 53 kernel in 1 layer.



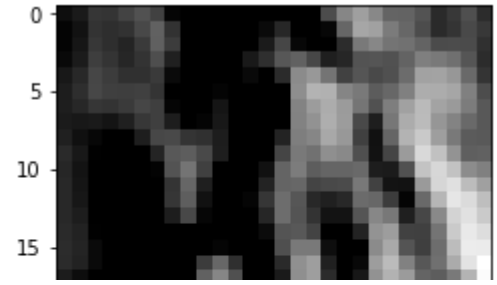
```
Probabilities of top 5 classes.
Actual class it belongs to: 28
Predicted class 28
Actual training image
```



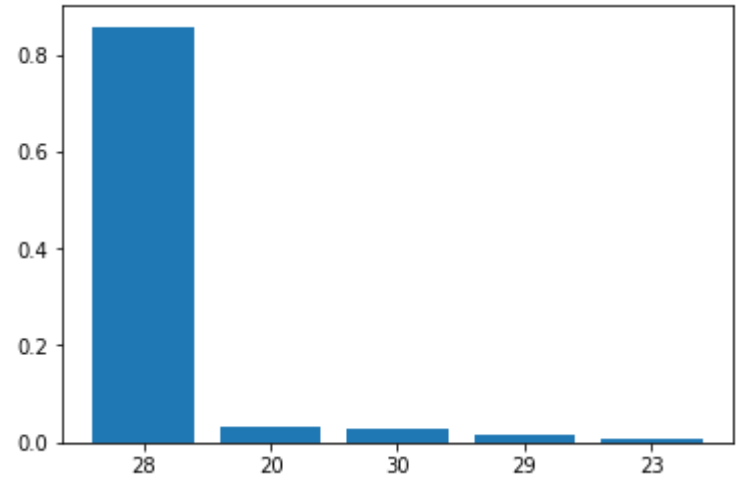

```
The kernel which activates/recognizes the shape: 22
```



Multiple kernel activations starting from 0



Activation for 22 kernel in 3 layer.

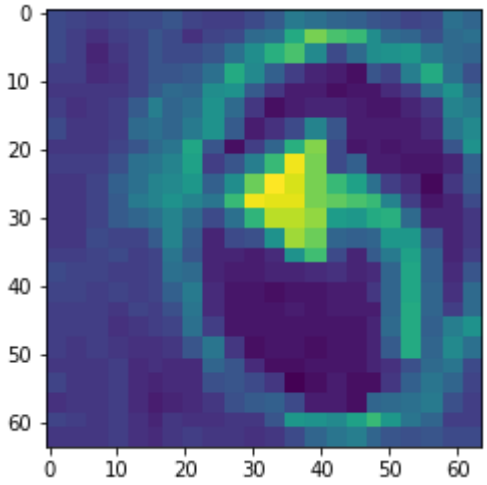


Probabilities of top 5 classes.

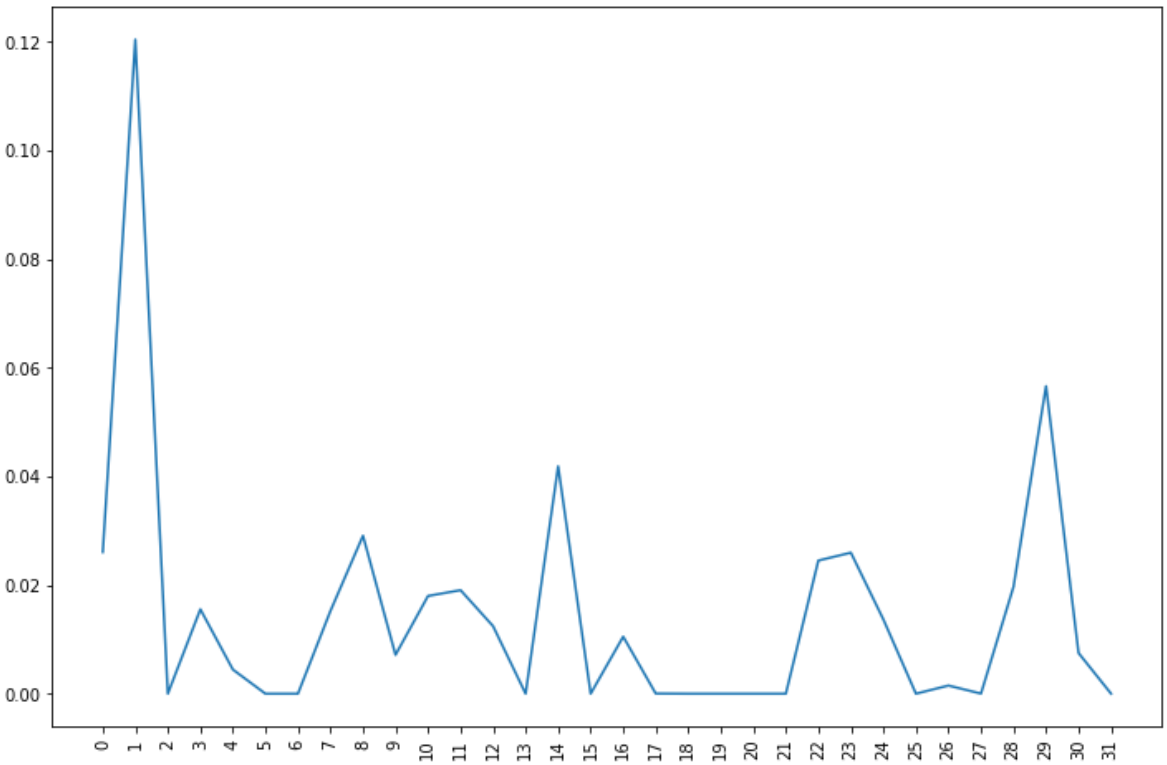
In [0]:

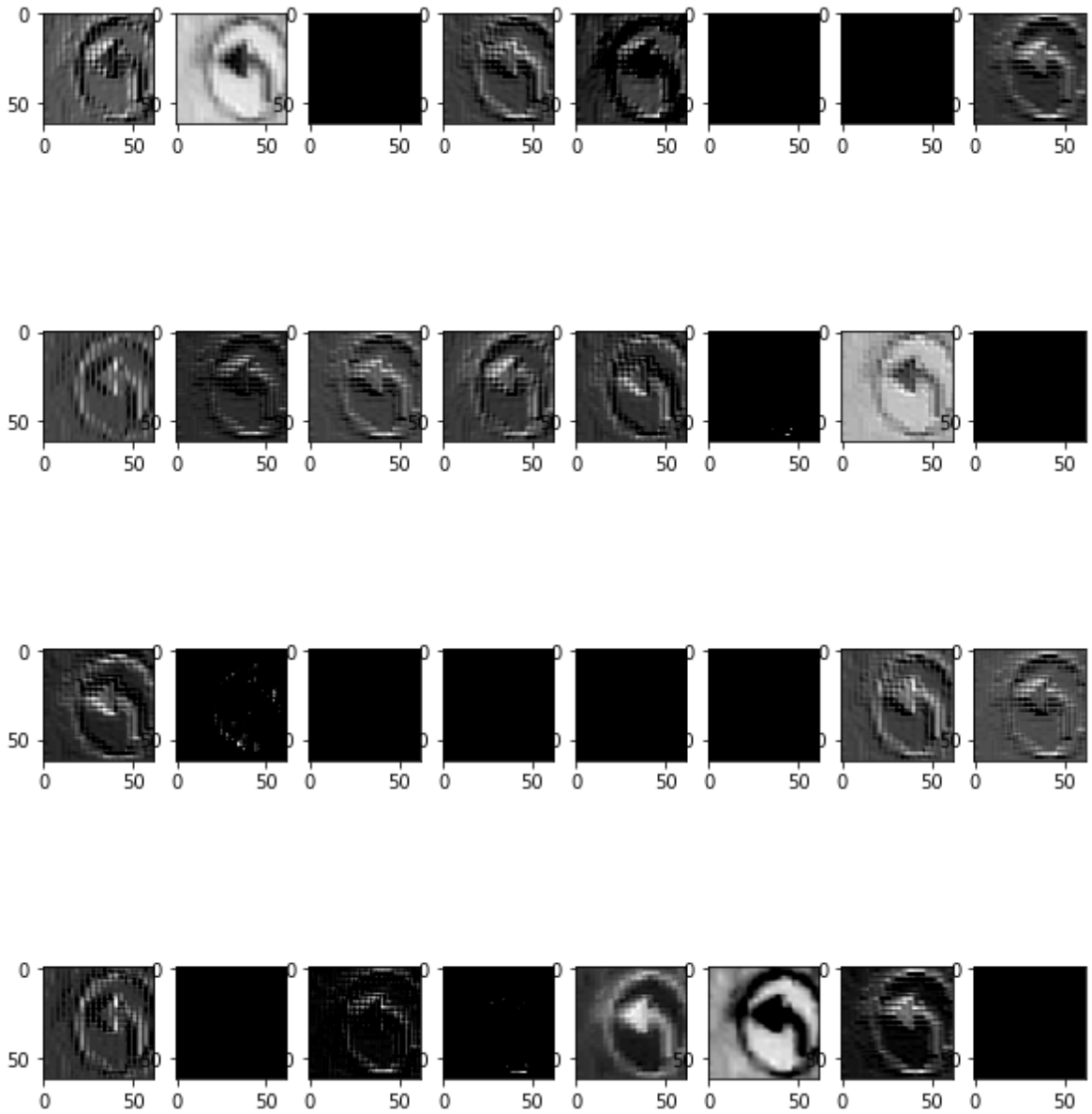
```
idx = 572
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

```
Actual class it belongs to: 34
Predicted class 34
Actual training image
```

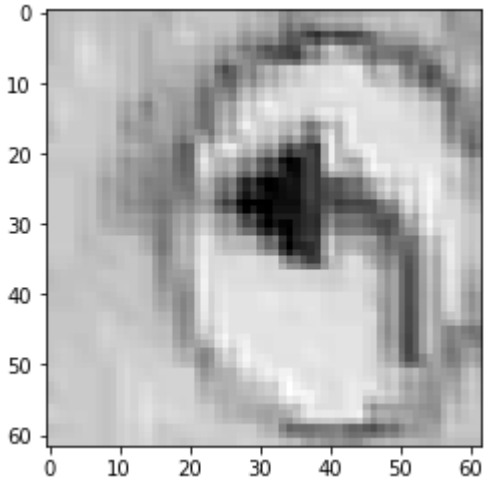



```
The kernel which activates/recognizes the shape: 1
```

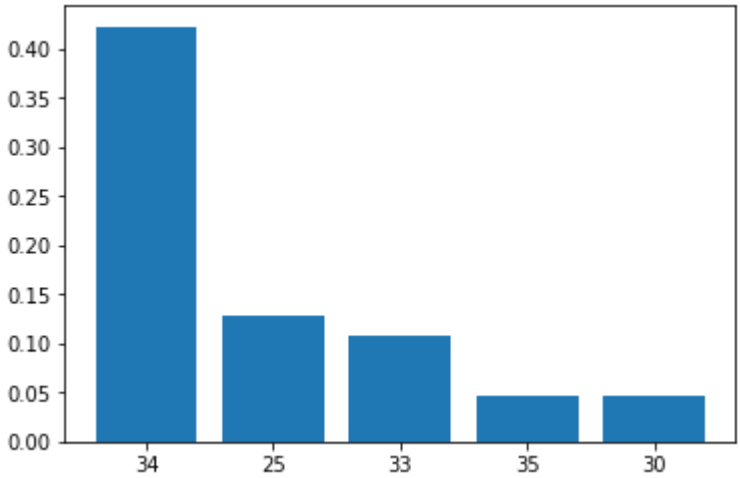




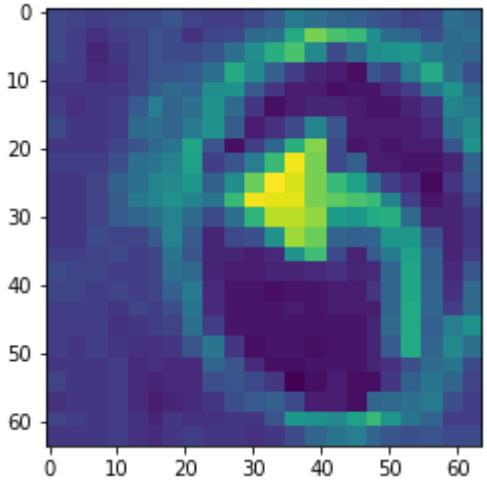
Multiple kernel activations starting from 0



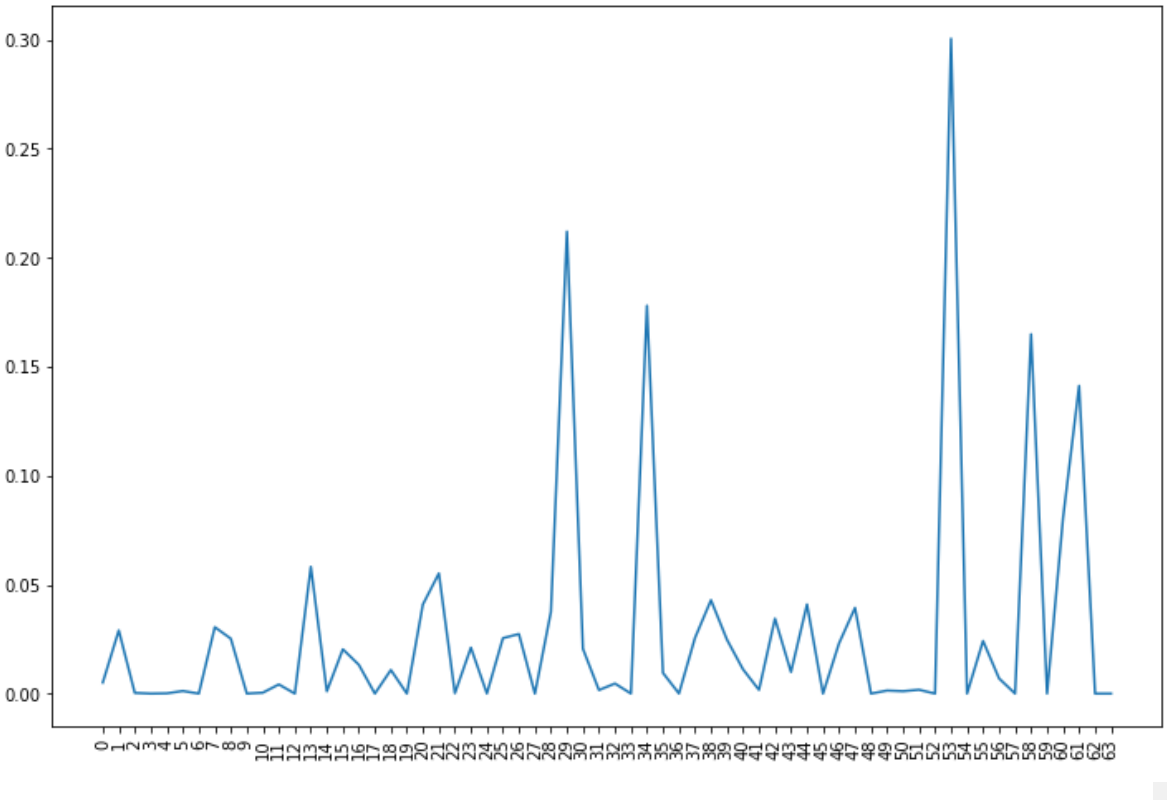
Activation for 1 kernel in 0 layer.

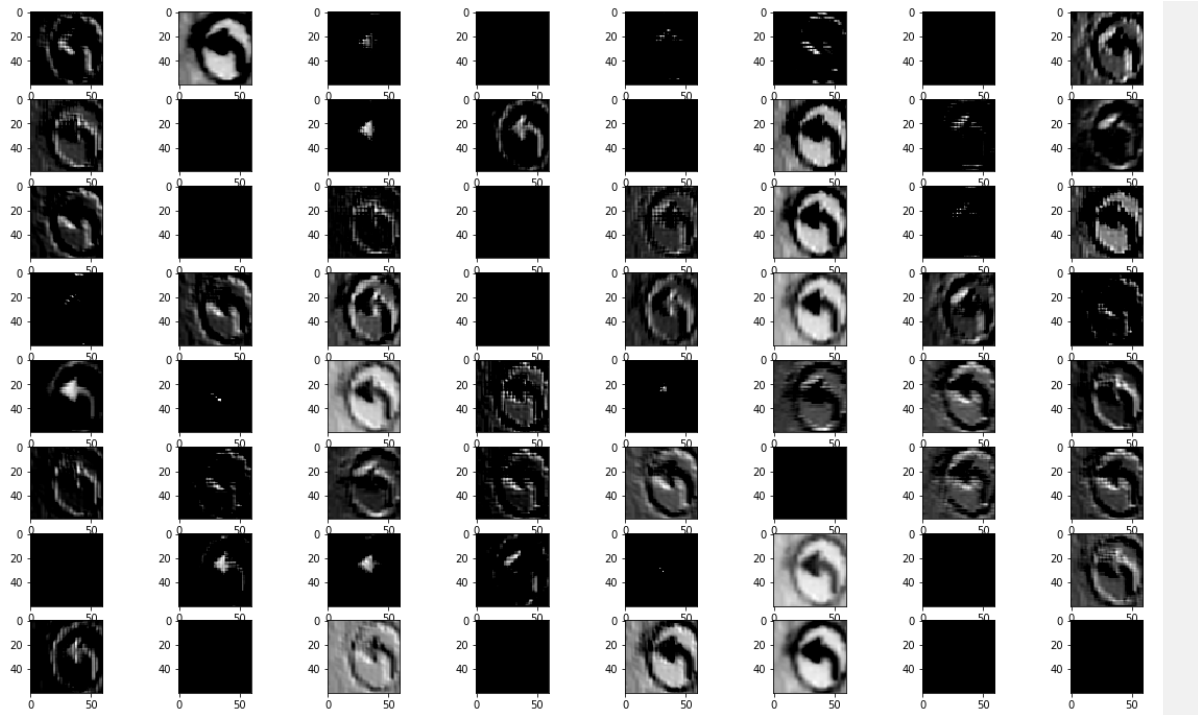



```
Probabilities of top 5 classes.
Actual class it belongs to: 34
Predicted class 34
Actual training image
```

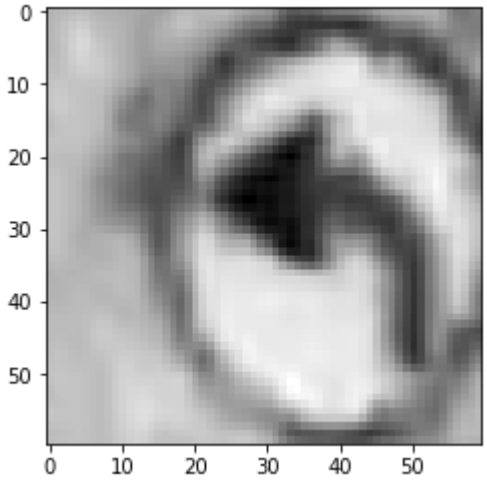



```
The kernel which activates/recognizes the shape: 53
```

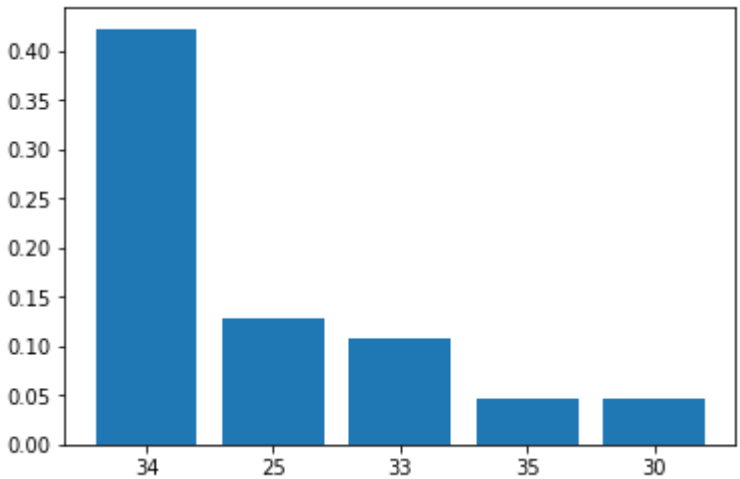




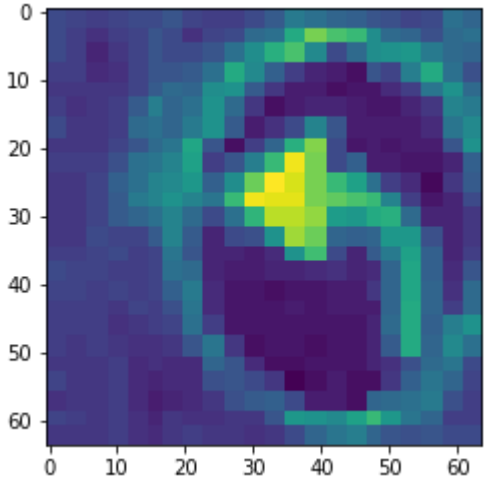
Multiple kernel activations starting from 0



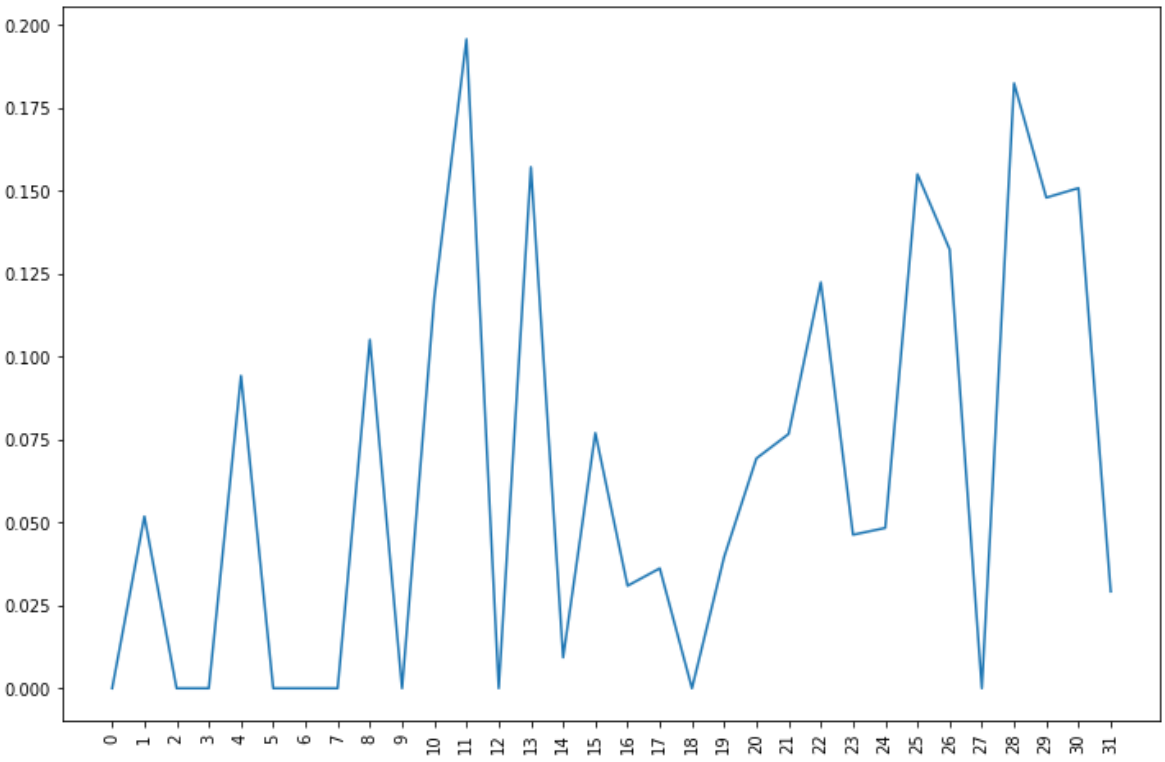
Activation for 53 kernel in 1 layer.

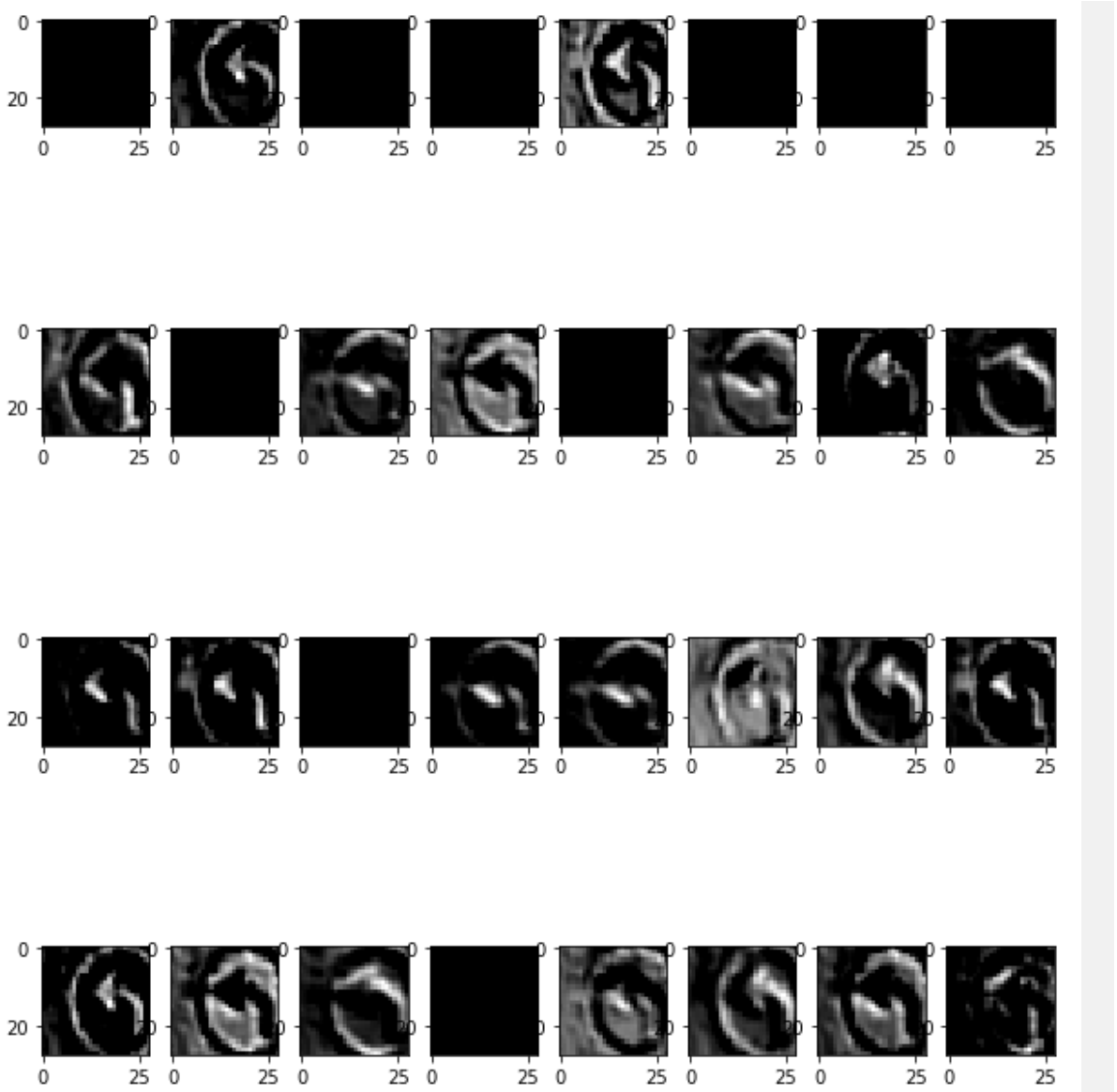


```
Probabilities of top 5 classes.
Actual class it belongs to: 34
Predicted class 34
Actual training image
```

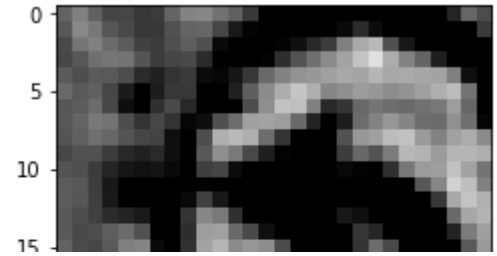



```
The kernel which activates/recognizes the shape: 11
```

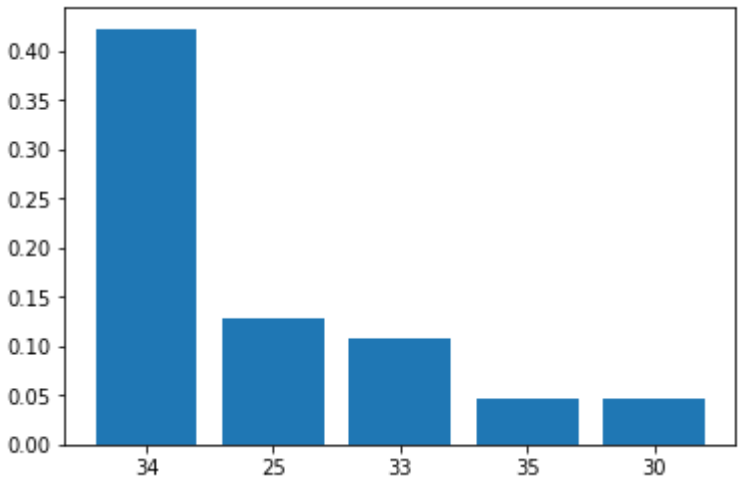




Multiple kernel activations starting from 0



Activation for 11 kernel in 3 layer.

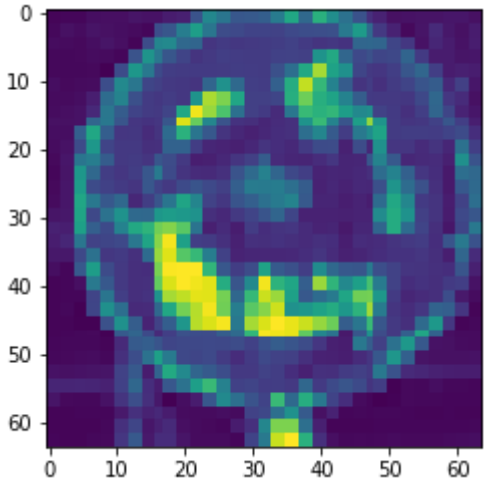


Probabilities of top 5 classes.

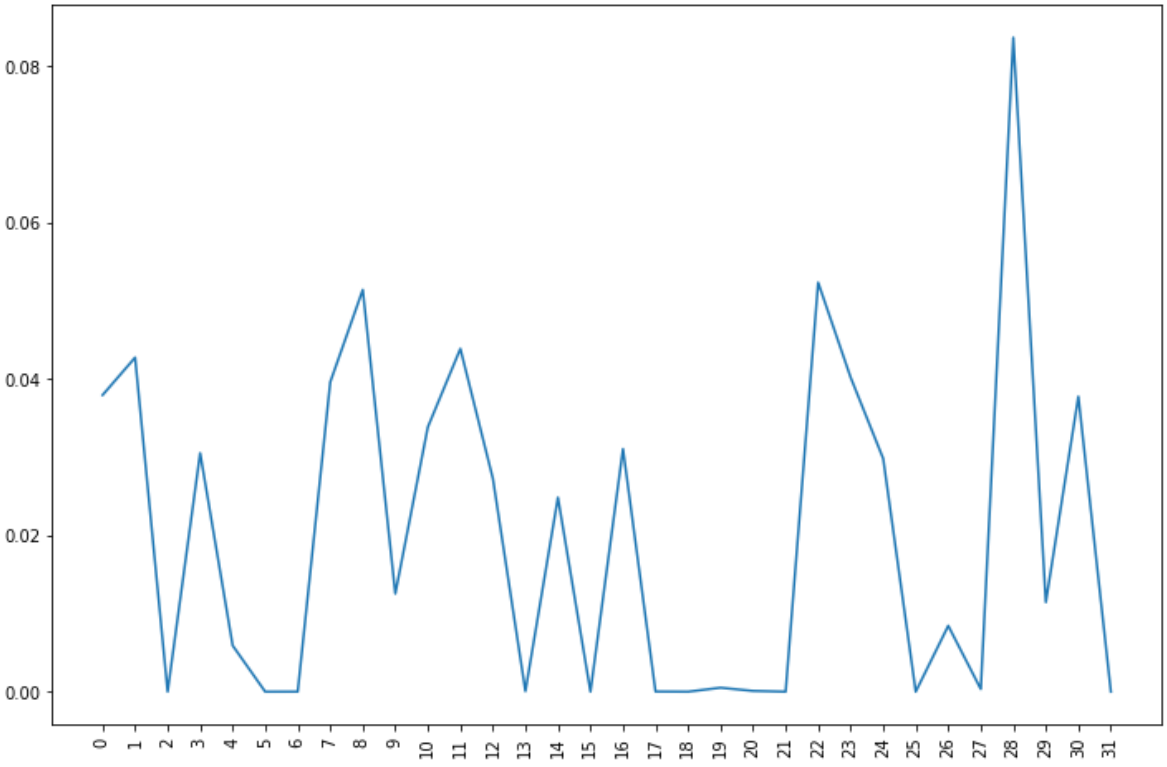
In [0]:

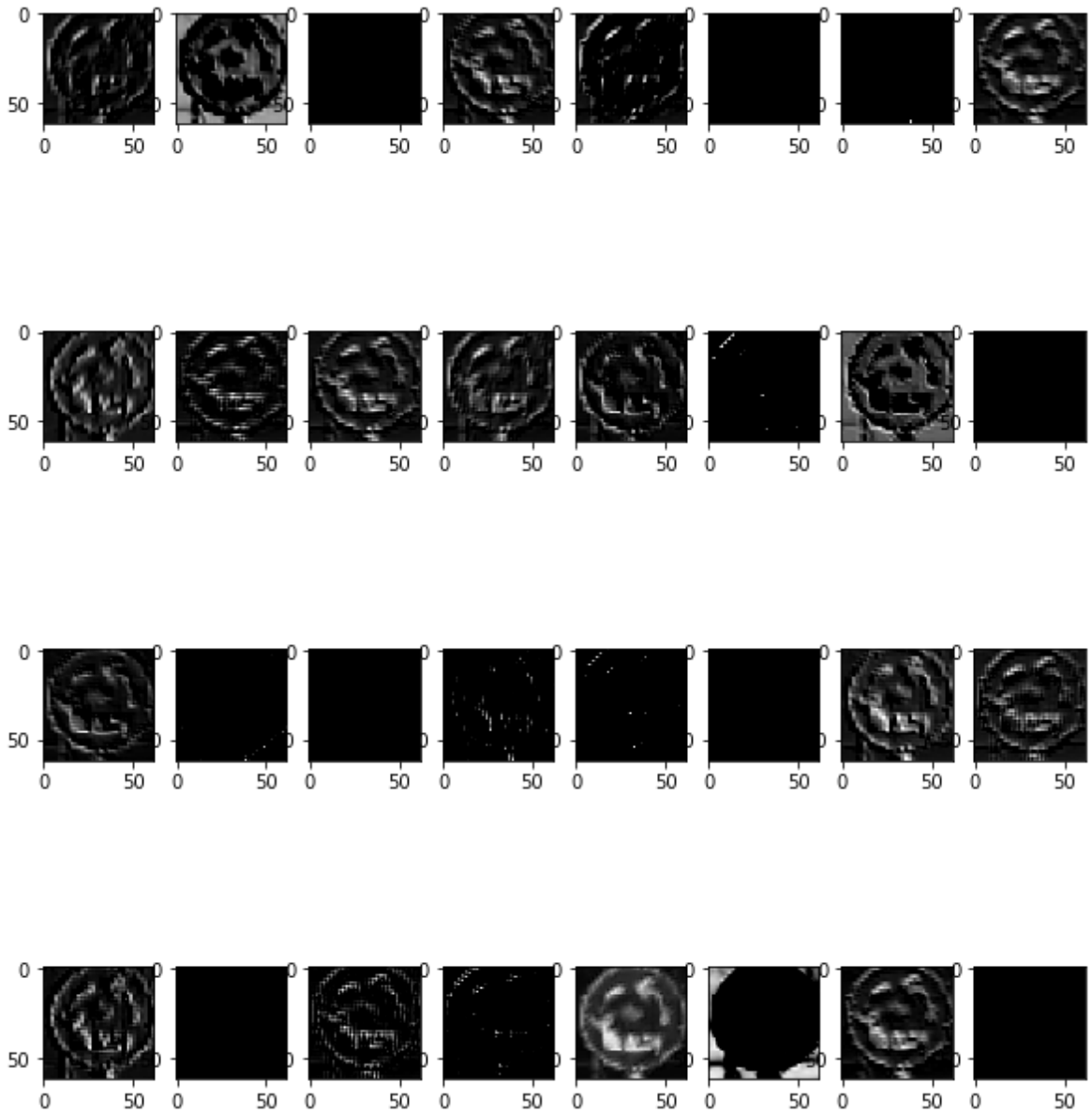
```
idx = 151
activations = activation_model.predict(x_test[idx].reshape(1, 64, 64, 1))
call_utilities_features(idx, activations, 8, 4, 0, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 8, 1, y_true, y_prediction)
call_utilities_features(idx, activations, 8, 4, 3, y_true, y_prediction)
```

```
Actual class it belongs to: 40
Predicted class 2
Actual training image
```

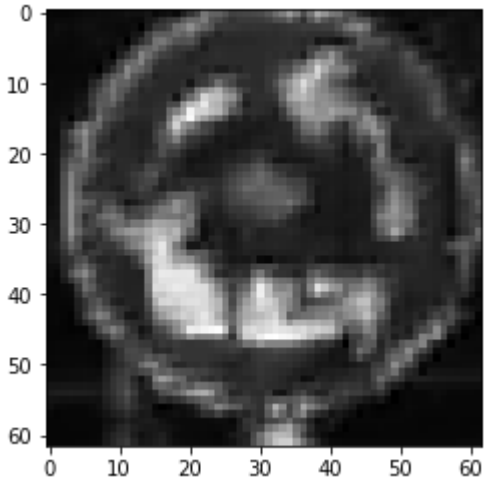



```
The kernel which activates/recognizes the shape: 28
```

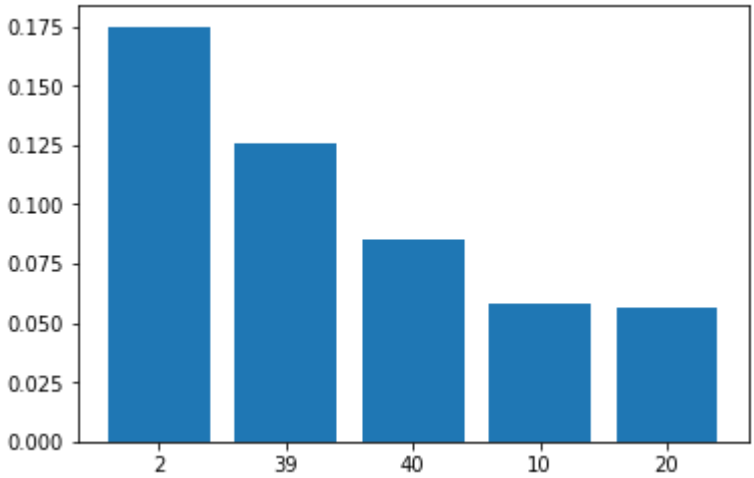




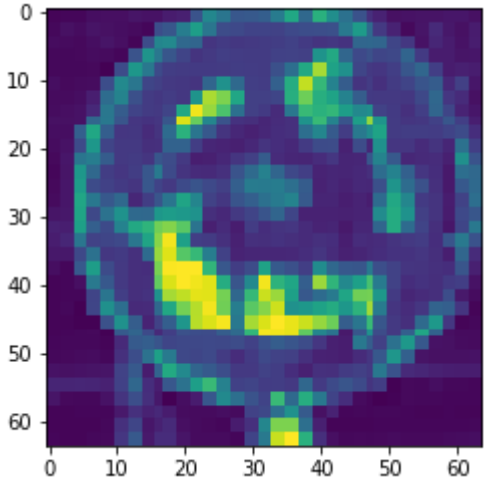
Multiple kernel activations starting from 0



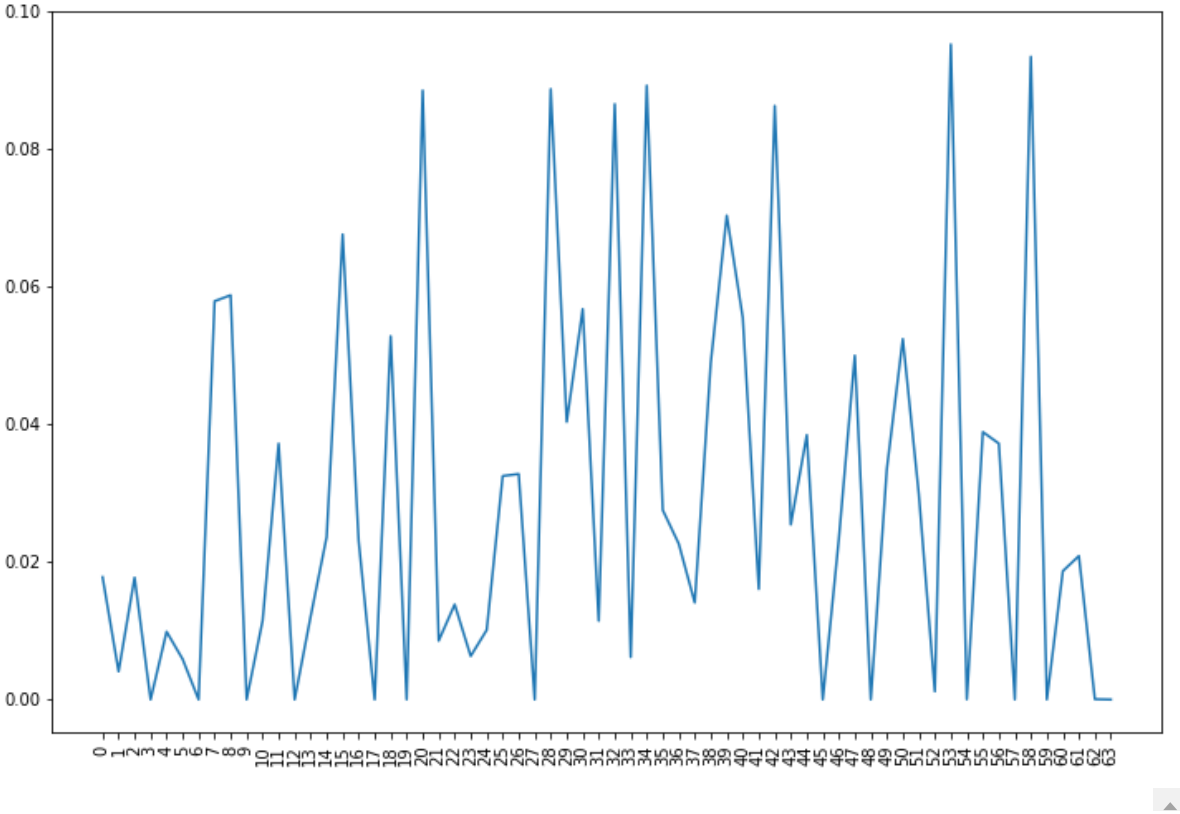
Activation for 28 kernel in 0 layer.

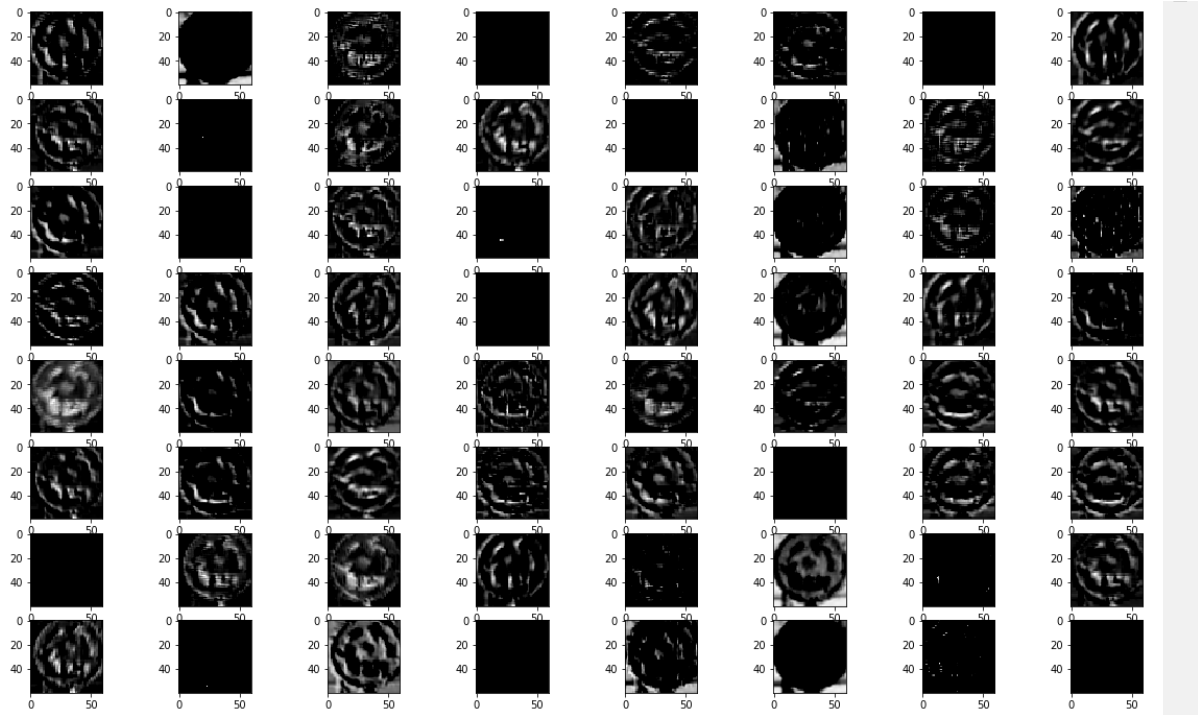



```
Probabilities of top 5 classes.
Actual class it belongs to: 40
Predicted class 2
Actual training image
```

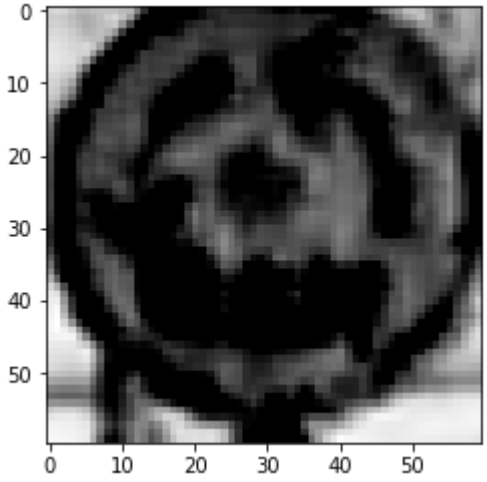



```
The kernel which activates/recognizes the shape: 53
```

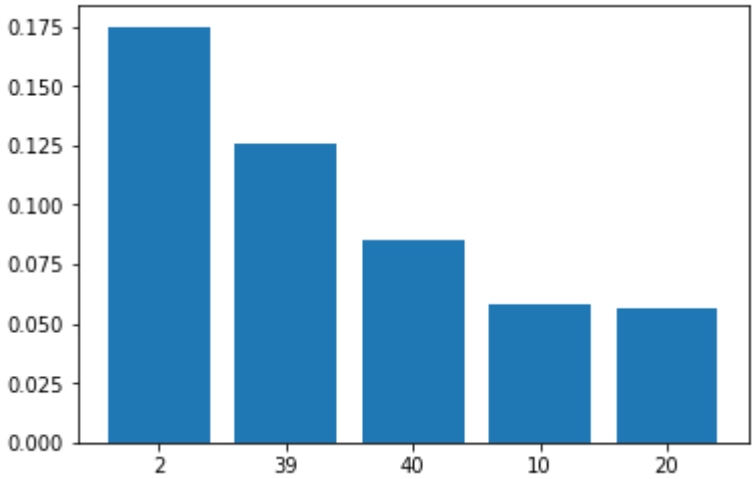




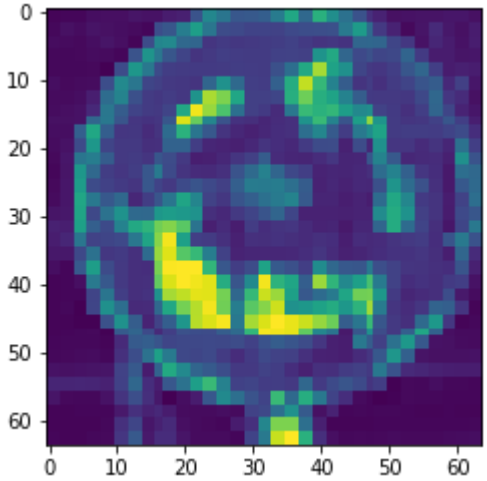
Multiple kernel activations starting from 0



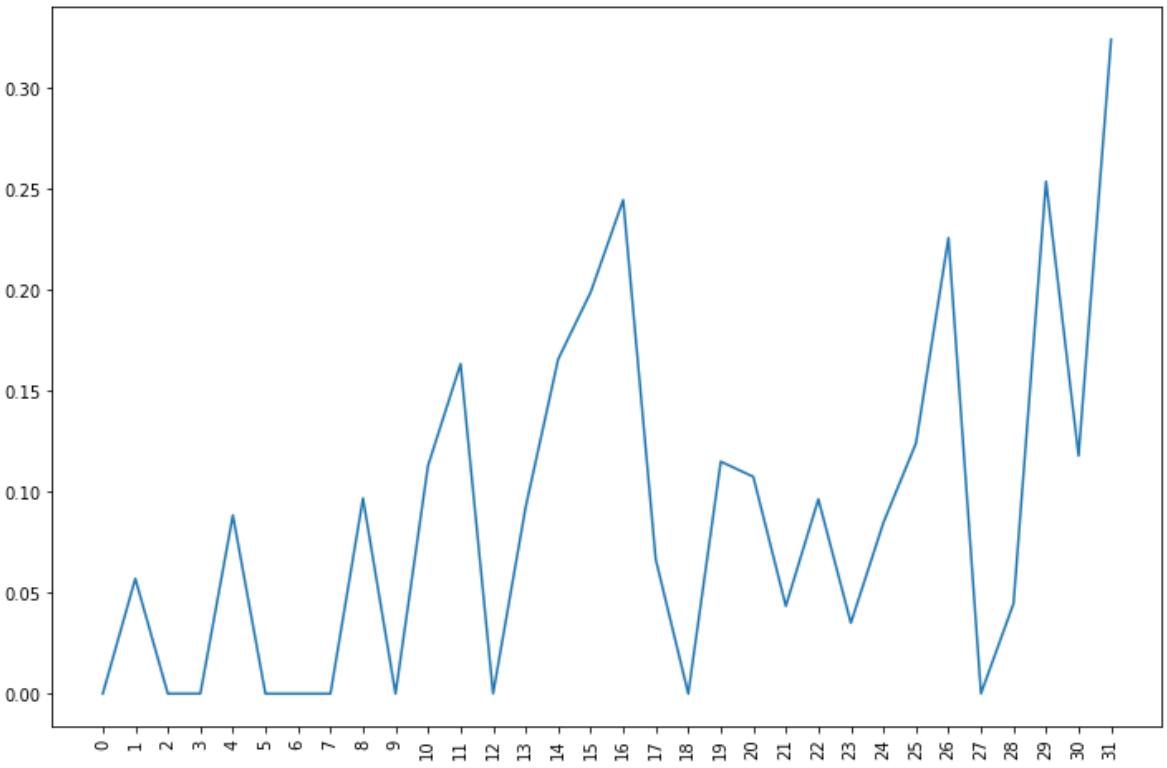
Activation for 53 kernel in 1 layer.

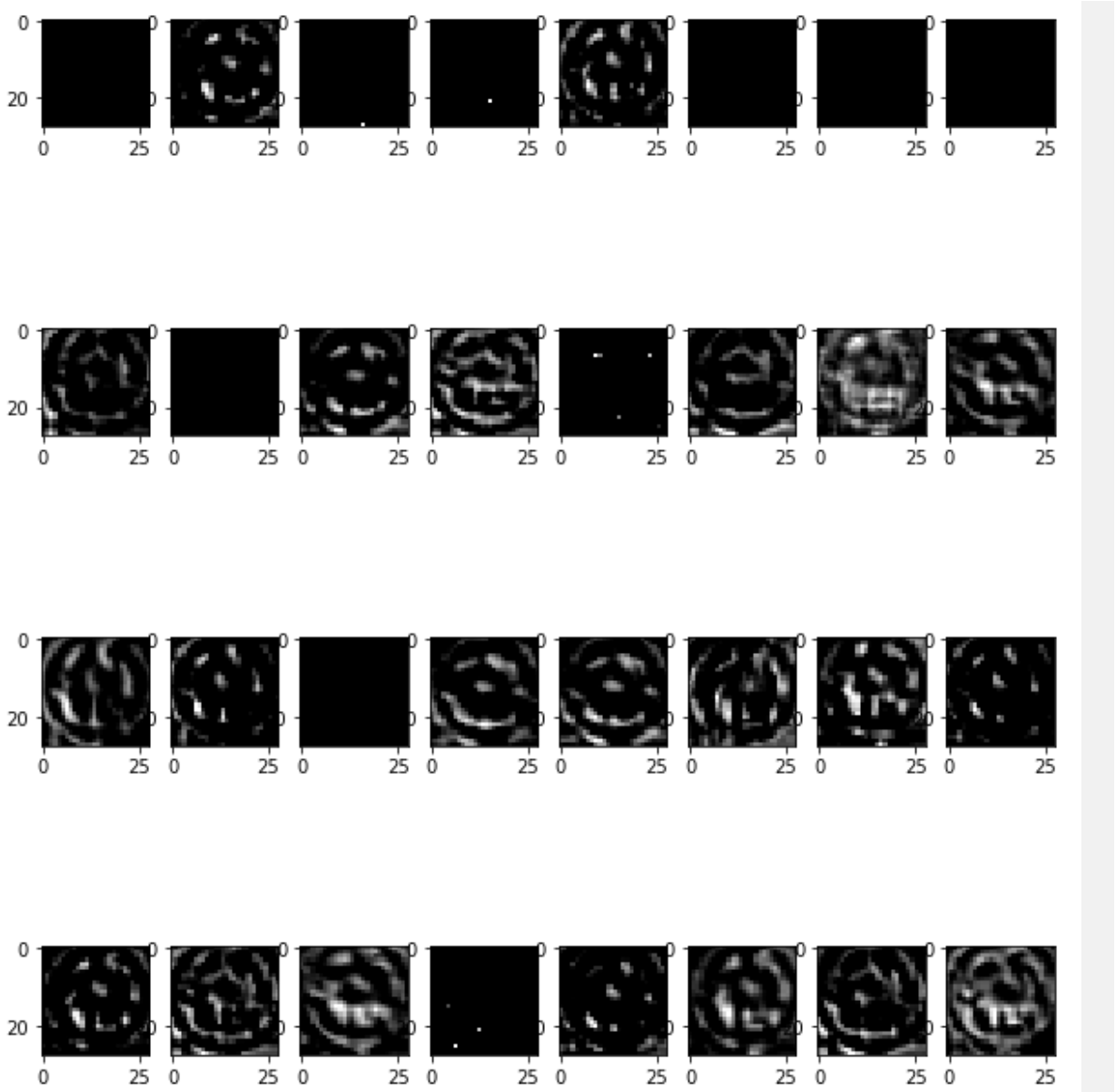


```
Probabilities of top 5 classes.
Actual class it belongs to: 40
Predicted class 2
Actual training image
```

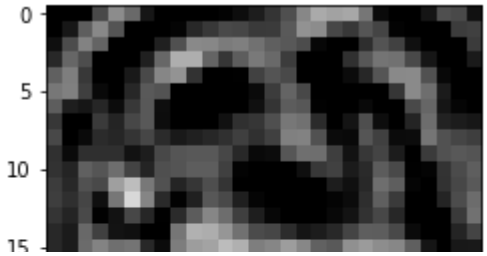



```
The kernel which activates/recognizes the shape: 31
```

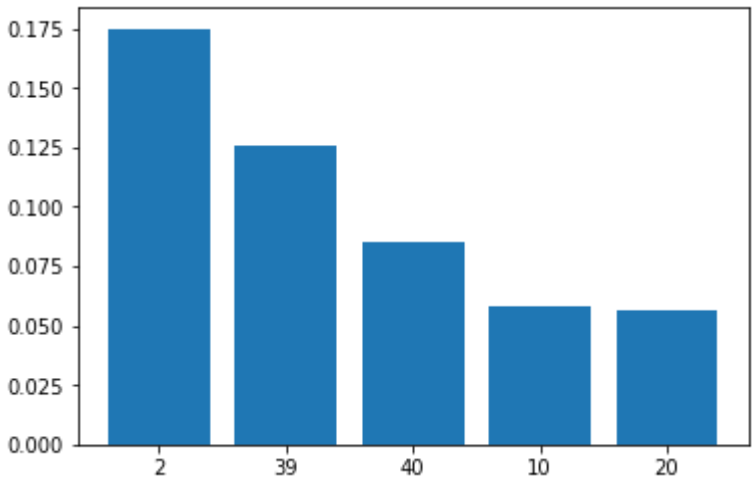




Multiple kernel activations starting from 0



Activation for 31 kernel in 3 layer.



Probabilities of top 5 classes.

In [0]:

```
actual_class = []
index_image = []
predicted_class = []
for idx, im in enumerate(y_true):
    if im != np.argmax(y_prediction[idx]):
        actual_class.append(im)
        predicted_class.append(np.argmax(y_prediction[idx]))
        index_image.append(idx)
np.array(index_image)
```

Out[132]:

```
array([   41,    80,   151,   163,   592,   621,   657,   696,   836,
         981,  1009,  1053,  1097,  1148,  1193,  1204,  1210,  1314,
        1326,  1391,  1395,  1485,  1500,  1720,  1748,  1750,  1755,
        1769,  1840,  1852,  1880,  1915,  1926,  1993,  2165,  2290,
        2361,  2465,  2516,  2873,  2895,  2917,  2925,  2963,  2970,
        2978,  3055,  3450,  3510,  3592,  3669,  3909,  3950,  3962,
        4083,  4123,  4157,  4162,  4316,  4475,  4743,  4921,  4926,
        5051,  5089,  5240,  5491,  5607,  5666,  5734,  5755,  5818,
        6102,  6166,  6175,  6384,  6446,  6464,  6795,  6806,  6867,
        6985,  6995,  7156,  7269,  7358,  7377,  7378,  7423,  7481,
        7520,  7521,  7570,  7626,  7678,  7725,  7796,  7870,  7871,
        7914,  8047,  8200,  8207,  8218,  8346,  8348,  8412,  8466,
        8492,  8510,  8542,  8569,  8578,  8598,  8675,  8723,  8735,
        8739,  8767,  8970,  9518,  9749,  9771,  9873,  9973, 10073,
       10098, 10125, 10381, 10589, 10882, 10917, 11042, 11090, 11167,
       11211, 11253, 11301, 11333, 11361, 11366, 11404, 11527, 11569,
       11653])
```

[5.7] RetinaNet - Installation and training

In [0]:

```
!git clone https://github.com/fizyr/keras-retinanet.git
```

```
Cloning into 'keras-retinanet'...
remote: Enumerating objects: 4750, done.
remote: Total 4750 (delta 0), reused 0 (delta 0), pack-reused 4750
Receiving objects: 100% (4750/4750), 13.04 MiB | 14.13 MiB/s, done.
Resolving deltas: 100% (3165/3165), done.
```

In [0]:

```
!pip install keras-retinanet/
```

```
Processing ./keras-retinanet
Requirement already satisfied: keras in /usr/local/lib/python3.6/dist-packag
es (from keras-retinanet==0.5.1) (2.2.4)
Collecting keras-resnet (from keras-retinanet==0.5.1)
  Downloading https://files.pythonhosted.org/packages/76/d4/a35cbd07381139dd
a4db42c81b88c59254faac026109022727b45b31bcad/keras-resnet-0.2.0.tar.gz (http
s://files.pythonhosted.org/packages/76/d4/a35cbd07381139dda4db42c81b88c59254
faac026109022727b45b31bcad/keras-resnet-0.2.0.tar.gz)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages
(from keras-retinanet==0.5.1) (1.12.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packag
es (from keras-retinanet==0.5.1) (1.3.0)
Requirement already satisfied: cython in /usr/local/lib/python3.6/dist-packa
ges (from keras-retinanet==0.5.1) (0.29.10)
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packa
ges (from keras-retinanet==0.5.1) (4.3.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dis
t-packages (from keras-retinanet==0.5.1) (3.4.5.20)
Requirement already satisfied: progressbar2 in /usr/local/lib/python3.6/dist
-packages (from keras-retinanet==0.5.1) (3.38.0)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist
-packages (from keras->keras-retinanet==0.5.1) (1.16.4)
Requirement already satisfied: keras-applications>=1.0.6 in /usr/local/lib/p
ython3.6/dist-packages (from keras->keras-retinanet==0.5.1) (1.0.8)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packa
ges (from keras->keras-retinanet==0.5.1) (3.13)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/
python3.6/dist-packages (from keras->keras-retinanet==0.5.1) (1.1.0)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-package
s (from keras->keras-retinanet==0.5.1) (2.8.0)
Requirement already satisfied: olefile in /usr/local/lib/python3.6/dist-pack
ages (from Pillow->keras-retinanet==0.5.1) (0.46)
Requirement already satisfied: python-utils>=2.3.0 in /usr/local/lib/python
3.6/dist-packages (from progressbar2->keras-retinanet==0.5.1) (2.3.0)
Building wheels for collected packages: keras-retinanet, keras-resnet
  Building wheel for keras-retinanet (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/b2/9f/57/cb0305f6f5a41fc3c11a
d67b8cedfbe9127775b563337827ba
  Building wheel for keras-resnet (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/5f/09/a5/497a30fd9ad9964e98a1
254d1e164bcd1b8a5eda36197ecb3c
Successfully built keras-retinanet keras-resnet
Installing collected packages: keras-resnet, keras-retinanet
Successfully installed keras-resnet-0.2.0 keras-retinanet-0.5.1
```

In [0]:

```
import keras_retinanet
```

In [0]:

```
model = keras_retinanet.models.backbone('resnet50').retinanet(num_classes=43)
```

In [0]:

```
df.head()
```

Out[21]:

	Filename	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId
0	13_00000_00000.ppm	25	25	5	5	20	20	13
1	13_00000_00001.ppm	26	27	5	6	21	22	13
2	13_00000_00002.ppm	26	28	5	6	21	22	13
3	13_00000_00003.ppm	28	28	5	5	23	23	13
4	13_00000_00004.ppm	29	29	5	5	23	23	13

In [0]:

```
# this code snippet contains code for ppm to jpg conversion
# and resizing the image
!mkdir jpg_data
for idx, row in tqdm(df.iterrows(), total=df.shape[0]):
    im = Image.open('data/'+row['Filename'])
    im.save('jpg_data/'+row['Filename'].split(".")[0]+".jpg", format='jpeg')
!ls jpg_data
```

```
100%|██████████| 39209/39209 [00:28<00:00, 1356.14it/s]
```

In [0]:

```
tqdm.pandas(desc='filename_rename')
df['Filename'] = df['Filename'].progress_apply(lambda x: '/content/jpg_data/'+str(x))
df['Filename'] = df['Filename'].progress_apply(lambda x: x.split(".")[0]+".jpg")
```

```
filename_rename: 100%|██████████| 39209/39209 [00:00<00:00, 564208.84it/s]
filename_rename: 100%|██████████| 39209/39209 [00:00<00:00, 560937.27it/s]
```

In [0]:

```
df.head()
```

Out[24]:

	Filename	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId
0	/content/jpg_data/13_00000_00000.jpg	25	25	5	5	20	20	13
1	/content/jpg_data/13_00000_00001.jpg	26	27	5	6	21	22	13
2	/content/jpg_data/13_00000_00002.jpg	26	28	5	6	21	22	13
3	/content/jpg_data/13_00000_00003.jpg	28	28	5	5	23	23	13
4	/content/jpg_data/13_00000_00004.jpg	29	29	5	5	23	23	13

In [0]:

```
data =  df[['Filename', 'Roi.X1', 'Roi.Y1', 'Roi.X2', 'Roi.Y2', 'ClassId']].rename(columns=
data.head()
```

Out[25]:

	Filename	x1	y1	x2	y2	ClassId
0	/content/jpg_data/13_00000_00000.jpg	5	5	20	20	13
1	/content/jpg_data/13_00000_00001.jpg	5	6	21	22	13
2	/content/jpg_data/13_00000_00002.jpg	5	6	21	22	13
3	/content/jpg_data/13_00000_00003.jpg	5	5	23	23	13
4	/content/jpg_data/13_00000_00004.jpg	5	5	23	23	13

In [0]:

```
mapping_df = pd.DataFrame({'class_id':[i for i in range(43)], 'id':[i for i in range(43)]})
mapping_df.head()
```

Out[26]:

	class_id	id
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4

In [0]:

```
x_train, x_test, y_train, y_test = train_test_split(data, data['ClassId'], stratify=data['C
x_train.shape, x_test.shape
```

Out[27]:

```
((31367, 6), (7842, 6))
```

In [0]:

```
x_train.to_csv("train.csv", index=False, header=False)
x_test.to_csv("test.csv", index=False, header=False)
mapping_df.to_csv("mapping.csv", index=False, header=False)
```

In [0]:

```
x_train.head()
```

Out[29]:

	Filename	x1	y1	x2	y2	ClassId
13698	/content/jpg_data/12_00031_00018.jpg	5	5	51	51	12
35520	/content/jpg_data/5_00031_00001.jpg	6	5	22	23	5
14848	/content/jpg_data/12_00069_00028.jpg	8	9	84	86	12
1582	/content/jpg_data/13_00052_00022.jpg	6	5	54	48	13
32603	/content/jpg_data/18_00017_00024.jpg	6	5	59	53	18

In [0]:

```
!nvidia-smi
```

```
Fri Jun 21 04:39:29 2019
+-----------------------------------------------------------------------------
--+
| NVIDIA-SMI 418.67       Driver Version: 410.79       CUDA Version: 10.0
|
|-------------------------------+----------------------+----------------------
--+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. EC
C |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute
M. |
|===============================+======================+======================
==|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |
0 |
| N/A   28C    P8    28W / 149W |      0MiB / 11441MiB |      0%      Defaul
t |
+-------------------------------+----------------------+----------------------
--+

+-----------------------------------------------------------------------------
--+
| Processes:                                                       GPU Memor
y |
|  GPU       PID   Type   Process name                             Usage
|
|=============================================================================
==|
|  No running processes found
|
+-----------------------------------------------------------------------------
--+
```

In [0]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clien
t_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.co
m&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2
F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.co
m%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.re
adonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&respon
se_type=code (https://accounts.google.com/o/oauth2/auth?client_id=9473189898
03-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=
urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleap
is.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:
..........
Mounted at /content/drive
```

In [0]:

```
!ls /content/drive/My\ Drive/retinanet_model
```

```
resnet50_csv_04_new.h5  resnet50_csv_09.h5       resnet50_csv_20.h5
resnet50_csv_05_new.h5  resnet50_csv_10_naya.h5
```

Notes:

1. Training was performed in three phases because colab terminates session.
2. Training was stopped after 12th epoch because there was not much improvement in regression loss (improved only by 0.0012) and in classification loss (improved only by 0.0049).

In [0]:

```
!retinanet-train --epochs 5 csv /content/train.csv /content/mapping.csv
```

```
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0618 06:52:55.301011 140596749281152 deprecation_wrapper.py:119] From /us
r/local/lib/python3.6/dist-packages/keras_retinanet/bin/train.py:66: The n
ame tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto inst
ead.

W0618 06:52:55.301325 140596749281152 deprecation_wrapper.py:119] From /us
r/local/lib/python3.6/dist-packages/keras_retinanet/bin/train.py:68: The n
ame tf.Session is deprecated. Please use tf.compat.v1.Session instead.

2019-06-18 06:52:55.317231: I tensorflow/core/platform/profile_utils/cpu_u
tils.cc:94] CPU Frequency: 2300000000 Hz
2019-06-18 06:52:55.317491: I tensorflow/compiler/xla/service/service.cc:1
68] XLA service 0x18a0840 executing computations on platform Host. Device
s:
2019-06-18 06:52:55.317536: I tensorflow/compiler/xla/service/service.cc:1
75]   StreamExecutor device (0): <undefined>, <undefined>
2019-06-18 06:52:55.319916: I tensorflow/stream_executor/platform/default/
```

In [0]:

```
!cp /content/drive/My\ Drive/retinanet_model/resnet50_csv_10_naya.h5 /content/
```

In [0]:

```
!ls
```

```
data                                images           resnet50_csv_10_naya.h5
drive                               jpg_data         sample_data
GTSRB_Final_Training_Images.zip     keras-retinanet  test.csv
GTSRB_Final_Training_Images.zip.1   mapping.csv      train.csv
```

In [0]:

```
!retinanet-train --epochs 5 --weights /content/resnet50_csv_05_new.h5 csv /content/train.cs
```

```
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0620 04:02:20.347955 139847697983360 deprecation_wrapper.py:119] From /us
r/local/lib/python3.6/dist-packages/keras_retinanet/bin/train.py:66: The n
ame tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto inst
ead.

W0620 04:02:20.348261 139847697983360 deprecation_wrapper.py:119] From /us
r/local/lib/python3.6/dist-packages/keras_retinanet/bin/train.py:68: The n
ame tf.Session is deprecated. Please use tf.compat.v1.Session instead.

2019-06-20 04:02:20.398283: I tensorflow/core/platform/profile_utils/cpu_u
tils.cc:94] CPU Frequency: 2300000000 Hz
2019-06-20 04:02:20.400691: I tensorflow/compiler/xla/service/service.cc:1
68] XLA service 0x26e0840 executing computations on platform Host. Device
s:
2019-06-20 04:02:20.400782: I tensorflow/compiler/xla/service/service.cc:1
75]   StreamExecutor device (0): <undefined>, <undefined>
2019-06-20 04:02:20.409916: I tensorflow/stream_executor/platform/default/
```

In [0]:

```
!retinanet-train --epochs 2 --weights /content/resnet50_csv_10_naya.h5 csv /content/train.c
```

```
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0621 04:42:18.014223 140293684475776 deprecation_wrapper.py:119] From /us
r/local/lib/python3.6/dist-packages/keras_retinanet/bin/train.py:66: The n
ame tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto inst
ead.

W0621 04:42:18.014537 140293684475776 deprecation_wrapper.py:119] From /us
r/local/lib/python3.6/dist-packages/keras_retinanet/bin/train.py:68: The n
ame tf.Session is deprecated. Please use tf.compat.v1.Session instead.

2019-06-21 04:42:18.072847: I tensorflow/core/platform/profile_utils/cpu_u
tils.cc:94] CPU Frequency: 2300000000 Hz
2019-06-21 04:42:18.076454: I tensorflow/compiler/xla/service/service.cc:1
68] XLA service 0x1b80840 executing computations on platform Host. Device
s:
2019-06-21 04:42:18.076495: I tensorflow/compiler/xla/service/service.cc:1
75]   StreamExecutor device (0): <undefined>, <undefined>
2019-06-21 04:42:18.086039: I tensorflow/stream_executor/platform/default/
```

In [0]:

```
!cp /content/snapshots/resnet50_csv_02.h5 /content/drive/My\ Drive/retinanet_model/resnet50
```

In [0]:

```
# import keras_retinanet
from keras_retinanet import models
from keras_retinanet.utils.image import read_image_bgr, preprocess_image, resize_image
from keras_retinanet.utils.visualization import draw_box, draw_caption
from keras_retinanet.utils.colors import label_color
import os
import cv2
import time
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [0]:

```
import tensorflow as tf
import keras
def get_session():
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    return tf.Session(config=config)
```

```
Using TensorFlow backend.
```

In [0]:

```
keras.backend.tensorflow_backend.set_session(get_session())
```

In [38]:

```
!retinanet-convert-model /content/snapshots/resnet50_csv_02.h5 /content/resnet50_csv_12_lat
```

```
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0621 09:26:04.081387 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras_retinanet/bin/convert_model.py:40: T
he name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto in
stead.

W0621 09:26:04.081714 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras_retinanet/bin/convert_model.py:42: T
he name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

2019-06-21 09:26:04.098675: I tensorflow/core/platform/profile_utils/cpu_uti
ls.cc:94] CPU Frequency: 2300000000 Hz
2019-06-21 09:26:04.099034: I tensorflow/compiler/xla/service/service.cc:16
8] XLA service 0x20532c0 executing computations on platform Host. Devices:
2019-06-21 09:26:04.099081: I tensorflow/compiler/xla/service/service.cc:17
5]   StreamExecutor device (0): <undefined>, <undefined>
2019-06-21 09:26:04.123597: I tensorflow/stream_executor/platform/default/ds
o_loader.cc:42] Successfully opened dynamic library libcuda.so.1
2019-06-21 09:26:04.126984: E tensorflow/stream_executor/cuda/cuda_driver.c
c:318] failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device i
s detected
2019-06-21 09:26:04.127047: I tensorflow/stream_executor/cuda/cuda_diagnosti
cs.cc:169] retrieving CUDA diagnostic information for host: 0763a31fa421
2019-06-21 09:26:04.127066: I tensorflow/stream_executor/cuda/cuda_diagnosti
cs.cc:176] hostname: 0763a31fa421
2019-06-21 09:26:04.127135: I tensorflow/stream_executor/cuda/cuda_diagnosti
cs.cc:200] libcuda reported version is: 410.79.0
2019-06-21 09:26:04.127202: I tensorflow/stream_executor/cuda/cuda_diagnosti
cs.cc:204] kernel reported version is: 410.79.0
2019-06-21 09:26:04.127221: I tensorflow/stream_executor/cuda/cuda_diagnosti
cs.cc:310] kernel version seems to match DSO: 410.79.0
W0621 09:26:04.159101 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: T
he name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder in
stead.

W0621 09:26:04.189113 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4115:
The name tf.random_normal is deprecated. Please use tf.random.normal instea
d.

W0621 09:26:04.338859 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138:
The name tf.random_uniform is deprecated. Please use tf.random.uniform inste
ad.

W0621 09:26:04.361407 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1919:
The name tf.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fu
sed_batch_norm instead.

W0621 09:26:04.363541 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3976:
The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

W0621 09:26:05.738678 140085689849728 deprecation_wrapper.py:119] From /usr/
```

```
local/lib/python3.6/dist-packages/keras_retinanet/backend/tensorflow_backen
d.py:68: The name tf.image.resize_images is deprecated. Please use tf.image.
resize instead.

2019-06-21 09:26:07.789944: W tensorflow/compiler/jit/mark_for_compilation_p
ass.cc:1412] (One-time warning): Not using XLA:CPU for cluster because envva
r TF_XLA_FLAGS=--tf_xla_cpu_global_jit was not set.  If you want XLA:CPU, ei
ther set that envvar, or use experimental_jit_scope to enable XLA:CPU.  To c
onfirm that XLA is active, pass --vmodule=xla_compilation_cache=1 (as a prop
er command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=--xl
a_hlo_profile.
W0621 09:26:08.992736 140085689849728 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/optimizers.py:790: The name tf.trai
n.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

W0621 09:26:09.002739 140085689849728 deprecation.py:323] From /usr/local/li
b/python3.6/dist-packages/keras_retinanet/backend/tensorflow_backend.py:104:
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
```

In [39]:

```
# model_path = os.path.join('/content', 'resnet50_csv_20_converted.h5')
# load retinanet model
model = models.load_model('/content/resnet50_csv_12_latest.h5', backbone_name='resnet50')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0621 09:26:53.523947 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: T
he name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder in
stead.

W0621 09:26:53.594806 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4115:
The name tf.random_normal is deprecated. Please use tf.random.normal instea
d.

W0621 09:26:53.749905 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138:
The name tf.random_uniform is deprecated. Please use tf.random.uniform inste
ad.

W0621 09:26:53.778099 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1919:
The name tf.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fu
sed_batch_norm instead.

W0621 09:26:53.781766 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3976:
The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

W0621 09:26:53.785468 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4185:
The name tf.truncated_normal is deprecated. Please use tf.random.truncated_n
ormal instead.

W0621 09:26:55.235566 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras_retinanet/backend/tensorflow_backen
d.py:68: The name tf.image.resize_images is deprecated. Please use tf.image.
resize instead.

W0621 09:26:56.149650 140574185113472 deprecation.py:323] From /usr/local/li
b/python3.6/dist-packages/keras_retinanet/backend/tensorflow_backend.py:104:
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
W0621 09:26:58.582644 140574185113472 deprecation_wrapper.py:119] From /usr/
local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:174: T
he name tf.get_default_session is deprecated. Please use tf.compat.v1.get_de
fault_session instead.

/usr/local/lib/python3.6/dist-packages/keras/engine/saving.py:292: UserWarni
ng: No training configuration found in save file: the model was *not* compil
ed. Compile it manually.
  warnings.warn('No training configuration found in save file: '
```

In [40]:

```
labels_to_names = {i:i for i in range(43)}
labels_to_names
```

Out[40]:

```
{0: 0,
 1: 1,
 2: 2,
 3: 3,
 4: 4,
 5: 5,
 6: 6,
 7: 7,
 8: 8,
 9: 9,
 10: 10,
 11: 11,
 12: 12,
 13: 13,
 14: 14,
 15: 15,
 16: 16,
 17: 17,
 18: 18,
 19: 19,
 20: 20,
 21: 21,
 22: 22,
 23: 23,
 24: 24,
 25: 25,
 26: 26,
 27: 27,
 28: 28,
 29: 29,
 30: 30,
 31: 31,
 32: 32,
 33: 33,
 34: 34,
 35: 35,
 36: 36,
 37: 37,
 38: 38,
 39: 39,
 40: 40,
 41: 41,
 42: 42}
```

In [0]:

```
# load image
image = read_image_bgr('/content/jpg_data/25_00003_00029.jpg')

# copy to draw on
draw = image.copy()
draw = cv2.cvtColor(draw, cv2.COLOR_BGR2RGB)

# preprocess image for network
image = preprocess_image(image)
image, scale = resize_image(image)

# process image
start = time.time()
boxes, scores, labels = model.predict_on_batch(np.expand_dims(image, axis=0))
print("processing time: ", time.time() - start)

# correct for image scale
boxes /= scale
# print(boxes[0], scores[0], labels[0])
# visualize detections
for box, score, label in zip(boxes[0], scores[0], labels[0]):
    # scores are sorted so we can break
    if score < 0.6:
        break

    color = label_color(label)
    print("Label:", labels_to_names[label])
    b = box.astype(int)
    draw_box(draw, b, color=color)

    caption = "{} {:.3f}".format(labels_to_names[label], score)
    draw_caption(draw, b, caption)

plt.figure(figsize=(5, 5))
plt.axis('off')
plt.imshow(draw)
plt.show()
```

```
processing time:  0.3018159866333008
Label: 25
```

In [0]:

```
# load image
image = read_image_bgr('/content/jpg_data/18_00017_00024.jpg')

# copy to draw on
draw = image.copy()
draw = cv2.cvtColor(draw, cv2.COLOR_BGR2RGB)

# preprocess image for network
image = preprocess_image(image)
image, scale = resize_image(image)

# process image
start = time.time()
boxes, scores, labels = model.predict_on_batch(np.expand_dims(image, axis=0))
print("processing time: ", time.time() - start)

# correct for image scale
boxes /= scale
# print(boxes[0], scores[0], labels[0])
# visualize detections
for box, score, label in zip(boxes[0], scores[0], labels[0]):
    # scores are sorted so we can break
    if score < 0.6:
        break

    color = label_color(label)
    print("Label:", labels_to_names[label])
    b = box.astype(int)
    draw_box(draw, b, color=color)

    caption = "{} {:.3f}".format(labels_to_names[label], score)
    draw_caption(draw, b, caption)

plt.figure(figsize=(3, 3))
plt.axis('off')
plt.imshow(draw)
plt.show()
```

```
processing time:  0.3089919090270996
Label: 18
```

In [0]:

```
# load image
image = read_image_bgr('/content/jpg_data/12_00069_00028.jpg')

# copy to draw on
draw = image.copy()
draw = cv2.cvtColor(draw, cv2.COLOR_BGR2RGB)

# preprocess image for network
image = preprocess_image(image)
image, scale = resize_image(image)

# process image
start = time.time()
boxes, scores, labels = model.predict_on_batch(np.expand_dims(image, axis=0))
print("processing time: ", time.time() - start)

# correct for image scale
boxes /= scale
# print(boxes[0], scores[0], labels[0])
# visualize detections
for box, score, label in zip(boxes[0], scores[0], labels[0]):
    # scores are sorted so we can break
    if score < 0.7:
        break

    color = label_color(label)
    print("Label:", labels_to_names[label])
    b = box.astype(int)
    draw_box(draw, b, color=color)

    caption = "{} {:.3f}".format(labels_to_names[label], score)
    draw_caption(draw, b, caption)

plt.figure(figsize=(3, 3))
plt.axis('off')
plt.imshow(draw)
plt.show()
```

```
processing time:  0.30597877502441406
Label: 12
```

In [41]:

```
x_test.head()
```

Out[41]:

	Filename	x1	y1	x2	y2	ClassId
31936	/content/jpg_data/28_00013_00017.jpg	6	6	61	60	28
11009	/content/jpg_data/2_00066_00029.jpg	5	5	42	44	2
4143	/content/jpg_data/1_00040_00003.jpg	5	6	29	33	1
7857	/content/jpg_data/25_00011_00027.jpg	12	15	134	150	25
28430	/content/jpg_data/38_00060_00021.jpg	5	6	41	47	38

In [42]:

```
y_test = x_test['ClassId'].values.tolist()
len(y_test)
```

Out[42]:

```
7842
```

In [43]:

```
y_pred = []
for img in tqdm(x_test['Filename'].values, total=x_test.shape[0]):
    # load image
    image = read_image_bgr(img)

    # preprocess image for network
    image = preprocess_image(image)
    image, scale = resize_image(image)

    # process image
#     start = time.time()
    _, scores, labels = model.predict_on_batch(np.expand_dims(image, axis=0))
    y_pred.append(labels[0][np.argmax(scores[0])])
```

```
100%|██████████| 7842/7842 [39:12<00:00,  4.10it/s]
```

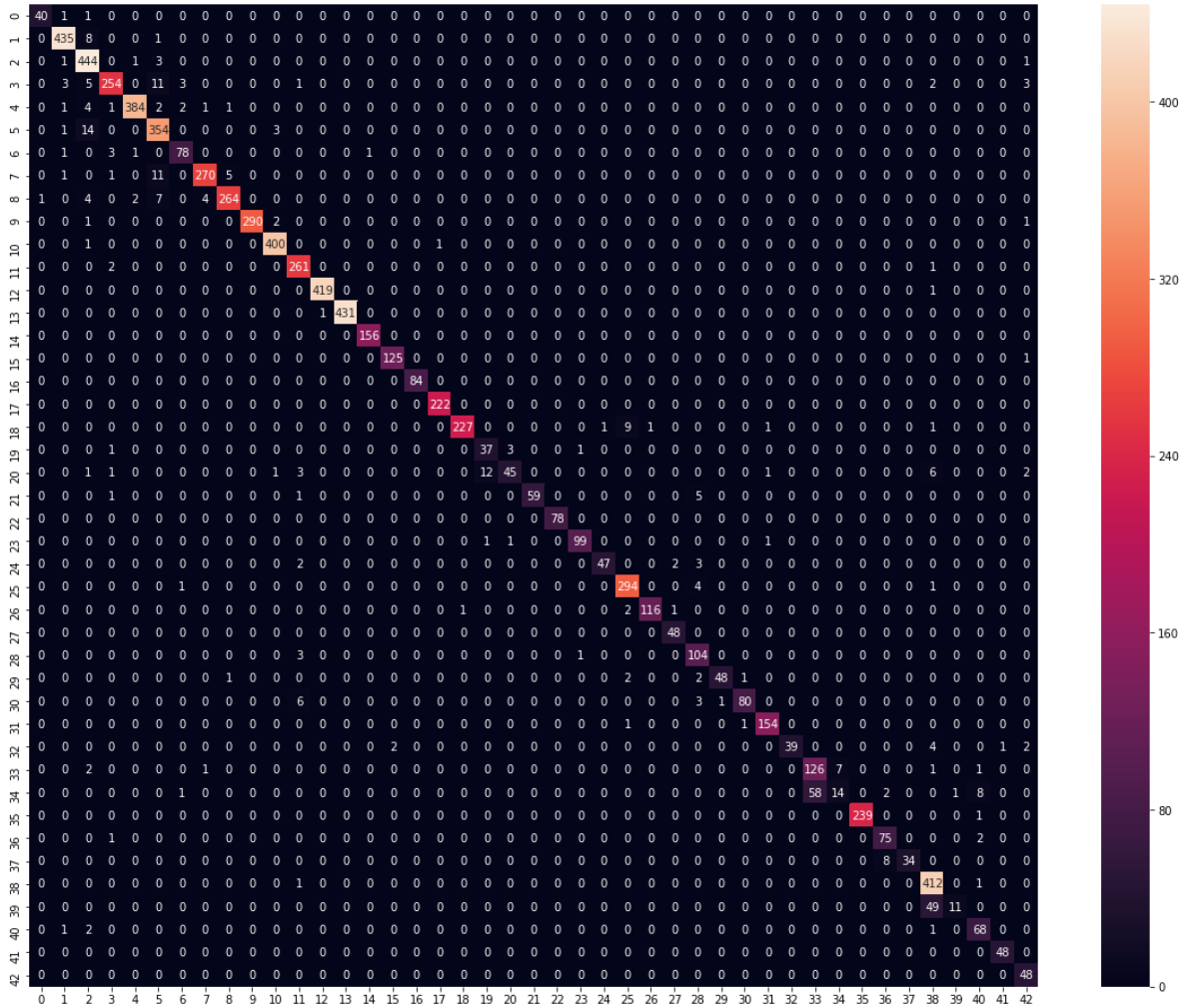
In [44]:

```
plt.figure(figsize=(20,16))
print("----------------------------------------------------------")
print("Micro F1 score", f1_score(y_test, y_pred, average='micro'))
print("----------------------------------------------------------")
confusion_mtx = confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_mtx, annot=True, fmt="d")
```

```
----------------------------------------------------------
Micro F1 score 0.9514154552410099
----------------------------------------------------------
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd99be9cef0>
```



[6] Conclusion

In [8]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model", "Test Micro F1 score"]
```

In [9]:

```
x.add_row(["Conv(8-3x3)-Conv(16-3x3)-MaxPool(2x2)-Dropout(0.75)-Dense(128)-Dropout(0.5)", 0
x.add_row(["Conv(16-3x3)-Conv(32-3x3)-MaxPool(2x2)-Dropout(0.75)-Dense(128)-Dropout(0.5)",
x.add_row(["Conv(32-4x4)-Conv(64-3x3)-MaxPool(2x2)-Dropout(0.75)-Dense(128)-Dropout(0.5)",
x.add_row(['''Conv(32-4x4)-Conv(64-3x3)-MaxPool(2x2)-Conv(32-4x4)
-Dropout(0.75)-Dense(128)-Dropout(0.5)''', 0.9892884468247896])
x.add_row(['''Conv(32-4x4)-Conv(64-3x3)-MaxPool(2x2)-Conv(32-4x4)-Dropout(0.75)
-MaxPool(2x2)-Conv(32-3x3)-Dense(256)-Dropout(0.5)-Dense(128)-Dropout(0.5)''', 0.9903085947
x.add_row(['''Conv(32-3x3)-Conv(64-3x3)-MaxPool(2x2)-Conv(32-3x3)-Dropout(0.75)
-MaxPool(2x2)-Conv(32-3x3)-Dense(256)-Dropout(0.5)-Dense(128)-Dropout(0.5) with Class Weigh
x.add_row(["RetinaNet", 0.9514154552410099])
print(x)
```

```
+-------------------------------------------------------------------------
-------------------+--------------------+
|                                             Model
| Test Micro F1 score |
+-------------------------------------------------------------------------
-------------------+--------------------+
|          Conv(8-3x3)-Conv(16-3x3)-MaxPool(2x2)-Dropout(0.75)-Dense(128)-Dr
opout(0.5)          |  0.9729660800816118 |
|          Conv(16-3x3)-Conv(32-3x3)-MaxPool(2x2)-Dropout(0.75)-Dense(128)-D
ropout(0.5)         |  0.9829125223157358 |
|          Conv(32-4x4)-Conv(64-3x3)-MaxPool(2x2)-Dropout(0.75)-Dense(128)-D
ropout(0.5)         |  0.9869081016747429 |
|                     Conv(32-4x4)-Conv(64-3x3)-MaxPool(2x2)-Conv(32-4x4)
|  0.9892884468247896 |
|                              -Dropout(0.75)-Dense(128)-Dropout(0.5)
|                     |
|               Conv(32-4x4)-Conv(64-3x3)-MaxPool(2x2)-Conv(32-4x4)-Dropout
(0.75)               |  0.9903085947462382 |
|           -MaxPool(2x2)-Conv(32-3x3)-Dense(256)-Dropout(0.5)-Dense(128)-Dr
opout(0.5)          |                     |
|               Conv(32-3x3)-Conv(64-3x3)-MaxPool(2x2)-Conv(32-3x3)-Dropout
(0.75)               |  0.9906486440533877 |
| -MaxPool(2x2)-Conv(32-3x3)-Dense(256)-Dropout(0.5)-Dense(128)-Dropout(0.5)
with Class Weights |                     |
|                                        RetinaNet
|  0.9514154552410099 |
+-------------------------------------------------------------------------
-------------------+--------------------+
```

Procedure

1. Downloaded the data and explored the folder structure involved.
2. Merged all the csv file into one dataframe (each class had a different csv file).
3. Appended the class id to file name and moved all images into one folder.
4. Plotted a bar graph to see the distribution of classes.
5. Cropped images and converted them into grey scale.
6. Converted all grey images into ndarray and split the data into train and test.
7. Wrote utility functions that help us better understand the neural network.
8. Built 8 different CNN models and plotted their confusion matrix using heat maps.
9. Trained RetinaNet which does both, predicts the location of traffic sign in the image and also classifies it into a class.
10. Recorded their micro F1 score and compared them in a table at the end.