Counting corners in polygons with a Convolutional Neural Network

In this notebook, I design a CNN network to count the corners of polygons in images from an archive. Images are found in the same working directory as this notebook along with the target labels.

The number of corners ranges from 3 to 10.

In [1]:
```
import numpy as np
import tensorflow as tf
import time

import numpy as np
import tensorflow as tf
import time

from tensorflow import keras
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Input, Activation, BatchNormaliza
from tensorflow.keras.layers import Conv2D, MaxPooling2D

from tensorflow.keras.optimizers import SGD, Adam, RMSprop, Nadam
from tensorflow.keras import backend as K
from tensorflow.keras import regularizers

from sklearn.metrics import *

import matplotlib
import matplotlib.pyplot as plt

import imageio.v2 as imageio
```

Define a function to load the polygon images

In [2]:
```
# functions for loading the images from the course Deep learning, Lund university (2022)
def loadImages(folder,trgFile,n):
    def load_pics(folder,n):
        imgs = []
        for i in range(n):
            img = imageio.imread(folder+"img_{:05}.png".format(i+1))
            ch = img[:,:,0]
            imgs.append(ch)
        return np.array(imgs)

    def load_labels(fn):
        return np.loadtxt(fn, usecols=0)

    pic = load_pics(folder+"/", n)
    ndata, width, height = pic.shape

    inp = (pic/np.float32(255)).reshape(n, width, height, 1)
    trg = load_labels(trgFile)
    trg = trg[0:n]

    return inp, trg, width, height


def loadData(nTrn, nTst):
```

```
    # Load data
    (trnInp, trnTrg, imgW, imgH) = loadImages("polyAll-trn", "polyAll-trn_trg.csv", nTrn)
    (tstInp, tstTrg, imgW, imgH) = loadImages("polyAll-tst", "polyAll-tst_trg.csv", nTst)

    if tf.keras.backend.image_data_format() == 'channels_first':
        trnInp = trnInp.reshape(trnInp.shape[0], 1, imgH, imgW)
        tstInp = tstInp.reshape(tstInp.shape[0], 1, imgH, imgW)
        input_shape = (1, imgH, imgW)
    else:
        trnInp = trnInp.reshape(trnInp.shape[0], imgH, imgW, 1)
        tstInp = tstInp.reshape(tstInp.shape[0], imgH, imgW, 1)
        input_shape = (imgH, imgW, 1)

    #print('trnInp shape:', trnInp.shape)
    #print('tstInp shape:', tstInp.shape)

    #trnTrg /= 10;
    #tstTrg /= 10;

    #sanna convert so that 0 means 3 and 7 means 10 etc
    trnTrg -= 3;
    tstTrg -= 3;

    return trnInp, trnTrg, tstInp, tstTrg, input_shape
```

Load and plot some example images

In [3]:
```
xplot, yplot, xplot2, yplot2, plotdata_shape = loadData(10,10)

print('Dimensions of images: ' , plotdata_shape)

plt.figure(1, figsize=(15,10))
plt.imshow(xplot[:10,:,:].swapaxes(0,1).reshape(100,10*100),cmap="gray")
plt.axis("off")
#plt.savefig('example_data')
print("Targets:")
print(yplot[:10])
print('acctual number of corners: ', yplot[:10]+3)
```

```
Dimensions of images:  (100, 100, 1)
Targets:
[2. 6. 3. 7. 3. 2. 2. 3. 1. 6.]
acctual number of corners:  [ 5.  9.  6. 10.  6.  5.  5.  6.  4.  9.]
```

- Load the dataset: training data and test data
- Make images binary (optional)
- Reserve 20% of the training data for validation

In [4]:
```
# Load the dataset
xtrain, ytrain, xtest, ytest, input_shape = loadData(5000,5000)

print('Shape of training data' , xtrain.shape)

#pre-proprocessing
# make all pixels < threshold black
#threshold = 0.1
#xtrain = 1.0 * (xtrain_in > threshold)
#xtest = 1.0 * (xtest_in > threshold)
```

```
# Reserve some samples for validation
val_param = int(xtrain.shape[0]*0.2)
print('Size of validation set: ', val_param)
xval = xtrain[0:val_param]
yval = ytrain[0:val_param]
xtrain = xtrain[val_param:]
ytrain = ytrain[val_param:]
```

```
Shape of training data (5000, 100, 100, 1)
Size of validation set:  1000
```

CNN architecture

In [5]:
```
model = Sequential([
    Conv2D(16, kernel_size = (3, 3),
           activation='relu',input_shape = input_shape, padding = 'same'),
    Conv2D(16, kernel_size = (3, 3), activation = 'relu', padding = 'same'),
    MaxPooling2D(pool_size = (2, 2)),
    Conv2D(32, kernel_size = (3, 3), activation = 'relu', padding = 'same'),
    Conv2D(32, kernel_size = (3, 3), activation = 'relu', padding = 'same'),
    MaxPooling2D(pool_size = (2, 2)),
    Conv2D(32, kernel_size = (3, 3), activation = 'relu', padding = 'same'),
    MaxPooling2D(pool_size = (2, 2)),
    Conv2D(64, kernel_size = (3, 3), activation = 'relu', padding = 'same'),
    MaxPooling2D(pool_size = (2, 2)),
    Flatten(),    #(dense layer needs a 1D input)
    Dense(128, activation='relu', kernel_regularizer = regularizers.l1_l2(0.0008, 0.008)),
    Dropout(0.5),
    Dense(8),            # 8 options output
    Activation('softmax')
])
```

Define training configuration (optimizer, loss, metrics)

In [6]:
```
# the loss function, optimization function
adam = Adam(learning_rate = 0.005)
#rms = RMSprop(learning_rate = 0.008)
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = adam, metrics = ['accuracy
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 100, 16)      160

 conv2d_1 (Conv2D)           (None, 100, 100, 16)      2320

 max_pooling2d (MaxPooling2D  (None, 50, 50, 16)       0
 )

 conv2d_2 (Conv2D)           (None, 50, 50, 32)        4640

 conv2d_3 (Conv2D)           (None, 50, 50, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 25, 25, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 25, 25, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 12, 12, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 12, 12, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 128)               295040

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 8)                 1032

 activation (Activation)     (None, 8)                 0

=================================================================
Total params: 340,184
Trainable params: 340,184
Non-trainable params: 0
_________________________________________________________________
```

Train the model

Train the model with the training data and validate with the validation data.

In [7]:
```
estimator = model.fit(xtrain, ytrain,
                      epochs = 150,
                      batch_size = 200, #64,#try something ~ between 20 and 200
                      verbose = 1,
                      validation_data = (xval, yval))


estimator.history
print('Accuracy: ', estimator.history['accuracy'][-1])
print(' Validation accuracy: ', estimator.history['val_accuracy'][-1])
```

```
Epoch 1/150
20/20 [==============================] - 12s 524ms/step - loss: 4.2896 - accuracy: 0.1797 - va
l_loss: 2.7125 - val_accuracy: 0.1960
Epoch 2/150
20/20 [==============================] - 9s 464ms/step - loss: 2.4891 - accuracy: 0.2323 - val
_loss: 2.2288 - val_accuracy: 0.2750
Epoch 3/150
20/20 [==============================] - 10s 504ms/step - loss: 2.2036 - accuracy: 0.2652 - va
l_loss: 2.0853 - val_accuracy: 0.3120
Epoch 4/150
20/20 [==============================] - 10s 495ms/step - loss: 2.1305 - accuracy: 0.2923 - va
l_loss: 2.0557 - val_accuracy: 0.3220
Epoch 5/150
20/20 [==============================] - 11s 530ms/step - loss: 2.0541 - accuracy: 0.3203 - va
l_loss: 1.9755 - val_accuracy: 0.3390
Epoch 6/150
20/20 [==============================] - 10s 494ms/step - loss: 2.0072 - accuracy: 0.3460 - va
l_loss: 1.9361 - val_accuracy: 0.3740
Epoch 7/150
20/20 [==============================] - 10s 515ms/step - loss: 1.9267 - accuracy: 0.3717 - va
l_loss: 1.8803 - val_accuracy: 0.3710
Epoch 8/150
20/20 [==============================] - 11s 555ms/step - loss: 1.9249 - accuracy: 0.3745 - va
l_loss: 1.7990 - val_accuracy: 0.4140
Epoch 9/150
20/20 [==============================] - 9s 468ms/step - loss: 1.8352 - accuracy: 0.3898 - val
_loss: 1.7444 - val_accuracy: 0.4130
Epoch 10/150
20/20 [==============================] - 10s 486ms/step - loss: 1.7870 - accuracy: 0.4160 - va
l_loss: 1.6824 - val_accuracy: 0.4670
Epoch 11/150
20/20 [==============================] - 10s 514ms/step - loss: 1.7763 - accuracy: 0.4378 - va
l_loss: 1.8572 - val_accuracy: 0.3890
Epoch 12/150
20/20 [==============================] - 10s 481ms/step - loss: 1.7322 - accuracy: 0.4395 - va
l_loss: 1.6018 - val_accuracy: 0.4880
Epoch 13/150
20/20 [==============================] - 10s 497ms/step - loss: 1.6005 - accuracy: 0.4805 - va
l_loss: 1.6087 - val_accuracy: 0.4510
Epoch 14/150
20/20 [==============================] - 9s 467ms/step - loss: 1.5664 - accuracy: 0.5052 - val
_loss: 1.5710 - val_accuracy: 0.5010
Epoch 15/150
20/20 [==============================] - 10s 496ms/step - loss: 1.5830 - accuracy: 0.5048 - va
l_loss: 1.6474 - val_accuracy: 0.4420
Epoch 16/150
20/20 [==============================] - 10s 499ms/step - loss: 1.5639 - accuracy: 0.5052 - va
l_loss: 1.5754 - val_accuracy: 0.4800
Epoch 17/150
20/20 [==============================] - 12s 593ms/step - loss: 1.5183 - accuracy: 0.5132 - va
l_loss: 1.4635 - val_accuracy: 0.5290
Epoch 18/150
20/20 [==============================] - 11s 536ms/step - loss: 1.4136 - accuracy: 0.5433 - va
l_loss: 1.4640 - val_accuracy: 0.5100
Epoch 19/150
20/20 [==============================] - 15s 753ms/step - loss: 1.5533 - accuracy: 0.5048 - va
l_loss: 1.5047 - val_accuracy: 0.5240
Epoch 20/150
20/20 [==============================] - 15s 748ms/step - loss: 1.4560 - accuracy: 0.5405 - va
l_loss: 1.3970 - val_accuracy: 0.5860
Epoch 21/150
20/20 [==============================] - 13s 677ms/step - loss: 1.3766 - accuracy: 0.5727 - va
l_loss: 1.3075 - val_accuracy: 0.6220
Epoch 22/150
20/20 [==============================] - 13s 631ms/step - loss: 1.3736 - accuracy: 0.5665 - va
l_loss: 1.3111 - val_accuracy: 0.6160
```

```
Epoch 23/150
20/20 [==============================] - 14s 698ms/step - loss: 1.3752 - accuracy: 0.5665 - va
l_loss: 1.4385 - val_accuracy: 0.5380
Epoch 24/150
20/20 [==============================] - 13s 641ms/step - loss: 1.3310 - accuracy: 0.6077 - va
l_loss: 1.3139 - val_accuracy: 0.6010
Epoch 25/150
20/20 [==============================] - 14s 704ms/step - loss: 1.2875 - accuracy: 0.6097 - va
l_loss: 1.2933 - val_accuracy: 0.6100
Epoch 26/150
20/20 [==============================] - 14s 691ms/step - loss: 1.3610 - accuracy: 0.5890 - va
l_loss: 1.5162 - val_accuracy: 0.5220
Epoch 27/150
20/20 [==============================] - 14s 682ms/step - loss: 1.3589 - accuracy: 0.5817 - va
l_loss: 1.3322 - val_accuracy: 0.5860
Epoch 28/150
20/20 [==============================] - 14s 688ms/step - loss: 1.3326 - accuracy: 0.5925 - va
l_loss: 1.3414 - val_accuracy: 0.6080
Epoch 29/150
20/20 [==============================] - 14s 682ms/step - loss: 1.2210 - accuracy: 0.6430 - va
l_loss: 1.2274 - val_accuracy: 0.6280
Epoch 30/150
20/20 [==============================] - 13s 639ms/step - loss: 1.1818 - accuracy: 0.6660 - va
l_loss: 1.2396 - val_accuracy: 0.6460
Epoch 31/150
20/20 [==============================] - 13s 650ms/step - loss: 1.2140 - accuracy: 0.6528 - va
l_loss: 1.2523 - val_accuracy: 0.6360
Epoch 32/150
20/20 [==============================] - 14s 694ms/step - loss: 1.2407 - accuracy: 0.6492 - va
l_loss: 1.2530 - val_accuracy: 0.6590
Epoch 33/150
20/20 [==============================] - 13s 635ms/step - loss: 1.2600 - accuracy: 0.6430 - va
l_loss: 1.3414 - val_accuracy: 0.5870
Epoch 34/150
20/20 [==============================] - 14s 674ms/step - loss: 1.2850 - accuracy: 0.6237 - va
l_loss: 1.2364 - val_accuracy: 0.6420
Epoch 35/150
20/20 [==============================] - 14s 683ms/step - loss: 1.1863 - accuracy: 0.6708 - va
l_loss: 1.2886 - val_accuracy: 0.6470
Epoch 36/150
20/20 [==============================] - 13s 647ms/step - loss: 1.2820 - accuracy: 0.6425 - va
l_loss: 1.3435 - val_accuracy: 0.6010
Epoch 37/150
20/20 [==============================] - 14s 715ms/step - loss: 1.2336 - accuracy: 0.6530 - va
l_loss: 1.1823 - val_accuracy: 0.6830
Epoch 38/150
20/20 [==============================] - 12s 608ms/step - loss: 1.1292 - accuracy: 0.6990 - va
l_loss: 1.1676 - val_accuracy: 0.6840
Epoch 39/150
20/20 [==============================] - 12s 616ms/step - loss: 1.1727 - accuracy: 0.6737 - va
l_loss: 1.3240 - val_accuracy: 0.5770
Epoch 40/150
20/20 [==============================] - 14s 698ms/step - loss: 1.1519 - accuracy: 0.6823 - va
l_loss: 1.2586 - val_accuracy: 0.6400
Epoch 41/150
20/20 [==============================] - 12s 600ms/step - loss: 1.1588 - accuracy: 0.6888 - va
l_loss: 1.4547 - val_accuracy: 0.5770
Epoch 42/150
20/20 [==============================] - 14s 688ms/step - loss: 1.2182 - accuracy: 0.6690 - va
l_loss: 1.2891 - val_accuracy: 0.6630
Epoch 43/150
20/20 [==============================] - 13s 634ms/step - loss: 1.1986 - accuracy: 0.6708 - va
l_loss: 1.2363 - val_accuracy: 0.6410
Epoch 44/150
20/20 [==============================] - 14s 725ms/step - loss: 1.1080 - accuracy: 0.7010 - va
l_loss: 1.2077 - val_accuracy: 0.6600
```

```
Epoch 45/150
20/20 [==============================] - 14s 693ms/step - loss: 1.3762 - accuracy: 0.5853 - va
l_loss: 1.4872 - val_accuracy: 0.5690
Epoch 46/150
20/20 [==============================] - 14s 687ms/step - loss: 1.1783 - accuracy: 0.6620 - va
l_loss: 1.2871 - val_accuracy: 0.6370
Epoch 47/150
20/20 [==============================] - 14s 707ms/step - loss: 1.1140 - accuracy: 0.6985 - va
l_loss: 1.1693 - val_accuracy: 0.6810
Epoch 48/150
20/20 [==============================] - 14s 685ms/step - loss: 1.0412 - accuracy: 0.7347 - va
l_loss: 1.1881 - val_accuracy: 0.6510
Epoch 49/150
20/20 [==============================] - 12s 597ms/step - loss: 1.0650 - accuracy: 0.7255 - va
l_loss: 1.2110 - val_accuracy: 0.6420
Epoch 50/150
20/20 [==============================] - 13s 642ms/step - loss: 1.0904 - accuracy: 0.7128 - va
l_loss: 1.2332 - val_accuracy: 0.6640
Epoch 51/150
20/20 [==============================] - 14s 702ms/step - loss: 1.0972 - accuracy: 0.7145 - va
l_loss: 1.2256 - val_accuracy: 0.6800
Epoch 52/150
20/20 [==============================] - 13s 660ms/step - loss: 1.0889 - accuracy: 0.7207 - va
l_loss: 1.1858 - val_accuracy: 0.6580
Epoch 53/150
20/20 [==============================] - 13s 669ms/step - loss: 1.0177 - accuracy: 0.7625 - va
l_loss: 1.1815 - val_accuracy: 0.6860
Epoch 54/150
20/20 [==============================] - 13s 648ms/step - loss: 1.0216 - accuracy: 0.7508 - va
l_loss: 1.1778 - val_accuracy: 0.7020
Epoch 55/150
20/20 [==============================] - 14s 685ms/step - loss: 1.1017 - accuracy: 0.7180 - va
l_loss: 1.3078 - val_accuracy: 0.6440
Epoch 56/150
20/20 [==============================] - 14s 719ms/step - loss: 1.0684 - accuracy: 0.7303 - va
l_loss: 1.2314 - val_accuracy: 0.6560
Epoch 57/150
20/20 [==============================] - 13s 647ms/step - loss: 1.0451 - accuracy: 0.7448 - va
l_loss: 1.2947 - val_accuracy: 0.6630
Epoch 58/150
20/20 [==============================] - 14s 688ms/step - loss: 1.0529 - accuracy: 0.7380 - va
l_loss: 1.2028 - val_accuracy: 0.6610
Epoch 59/150
20/20 [==============================] - 14s 718ms/step - loss: 1.0592 - accuracy: 0.7362 - va
l_loss: 1.1584 - val_accuracy: 0.6890
Epoch 60/150
20/20 [==============================] - 13s 649ms/step - loss: 1.0184 - accuracy: 0.7605 - va
l_loss: 1.2021 - val_accuracy: 0.6540
Epoch 61/150
20/20 [==============================] - 14s 713ms/step - loss: 1.0523 - accuracy: 0.7475 - va
l_loss: 1.4782 - val_accuracy: 0.6280
Epoch 62/150
20/20 [==============================] - 14s 696ms/step - loss: 1.1011 - accuracy: 0.7253 - va
l_loss: 1.2089 - val_accuracy: 0.6750
Epoch 63/150
20/20 [==============================] - 13s 645ms/step - loss: 1.1261 - accuracy: 0.7105 - va
l_loss: 1.2732 - val_accuracy: 0.6450
Epoch 64/150
20/20 [==============================] - 14s 678ms/step - loss: 1.0618 - accuracy: 0.7383 - va
l_loss: 1.2609 - val_accuracy: 0.6540
Epoch 65/150
20/20 [==============================] - 13s 663ms/step - loss: 1.0868 - accuracy: 0.7278 - va
l_loss: 1.2836 - val_accuracy: 0.6120
Epoch 66/150
20/20 [==============================] - 12s 610ms/step - loss: 1.0289 - accuracy: 0.7550 - va
l_loss: 1.1852 - val_accuracy: 0.6960
```

```
Epoch 67/150
20/20 [==============================] - 12s 602ms/step - loss: 0.9764 - accuracy: 0.7830 - va
l_loss: 1.2776 - val_accuracy: 0.6830
Epoch 68/150
20/20 [==============================] - 13s 668ms/step - loss: 0.9988 - accuracy: 0.7740 - va
l_loss: 1.2244 - val_accuracy: 0.6850
Epoch 69/150
20/20 [==============================] - 13s 670ms/step - loss: 1.0188 - accuracy: 0.7598 - va
l_loss: 1.1966 - val_accuracy: 0.6860
Epoch 70/150
20/20 [==============================] - 13s 649ms/step - loss: 1.0866 - accuracy: 0.7345 - va
l_loss: 1.1885 - val_accuracy: 0.6930
Epoch 71/150
20/20 [==============================] - 14s 690ms/step - loss: 1.0806 - accuracy: 0.7370 - va
l_loss: 1.2118 - val_accuracy: 0.6980
Epoch 72/150
20/20 [==============================] - 13s 624ms/step - loss: 0.9671 - accuracy: 0.7870 - va
l_loss: 1.2062 - val_accuracy: 0.7050
Epoch 73/150
20/20 [==============================] - 14s 697ms/step - loss: 0.9662 - accuracy: 0.7797 - va
l_loss: 1.3164 - val_accuracy: 0.6600
Epoch 74/150
20/20 [==============================] - 14s 697ms/step - loss: 1.1073 - accuracy: 0.7107 - va
l_loss: 1.2128 - val_accuracy: 0.6870
Epoch 75/150
20/20 [==============================] - 13s 630ms/step - loss: 1.0110 - accuracy: 0.7638 - va
l_loss: 1.2464 - val_accuracy: 0.6490
Epoch 76/150
20/20 [==============================] - 12s 607ms/step - loss: 0.9918 - accuracy: 0.7765 - va
l_loss: 1.2067 - val_accuracy: 0.6990
Epoch 77/150
20/20 [==============================] - 13s 654ms/step - loss: 0.9622 - accuracy: 0.7875 - va
l_loss: 1.2466 - val_accuracy: 0.6900
Epoch 78/150
20/20 [==============================] - 15s 752ms/step - loss: 0.9441 - accuracy: 0.8000 - va
l_loss: 1.2608 - val_accuracy: 0.6550
Epoch 79/150
20/20 [==============================] - 14s 672ms/step - loss: 0.9955 - accuracy: 0.7825 - va
l_loss: 1.2582 - val_accuracy: 0.6560
Epoch 80/150
20/20 [==============================] - 13s 655ms/step - loss: 1.0105 - accuracy: 0.7592 - va
l_loss: 1.2382 - val_accuracy: 0.6620
Epoch 81/150
20/20 [==============================] - 13s 653ms/step - loss: 1.0082 - accuracy: 0.7722 - va
l_loss: 1.4523 - val_accuracy: 0.5950
Epoch 82/150
20/20 [==============================] - 13s 635ms/step - loss: 1.1743 - accuracy: 0.6820 - va
l_loss: 1.3513 - val_accuracy: 0.6770
Epoch 83/150
20/20 [==============================] - 14s 697ms/step - loss: 1.0107 - accuracy: 0.7768 - va
l_loss: 1.2750 - val_accuracy: 0.6930
Epoch 84/150
20/20 [==============================] - 12s 620ms/step - loss: 0.9545 - accuracy: 0.8008 - va
l_loss: 1.2327 - val_accuracy: 0.6780
Epoch 85/150
20/20 [==============================] - 13s 657ms/step - loss: 0.9460 - accuracy: 0.8033 - va
l_loss: 1.2060 - val_accuracy: 0.7100
Epoch 86/150
20/20 [==============================] - 13s 671ms/step - loss: 0.9808 - accuracy: 0.7860 - va
l_loss: 1.3297 - val_accuracy: 0.6640
Epoch 87/150
20/20 [==============================] - 13s 646ms/step - loss: 0.9792 - accuracy: 0.7785 - va
l_loss: 1.2415 - val_accuracy: 0.6820
Epoch 88/150
20/20 [==============================] - 14s 696ms/step - loss: 0.9716 - accuracy: 0.7885 - va
l_loss: 1.2299 - val_accuracy: 0.7100
```

```
Epoch 89/150
20/20 [==============================] - 13s 675ms/step - loss: 0.9462 - accuracy: 0.7928 - va
l_loss: 1.1691 - val_accuracy: 0.6920
Epoch 90/150
20/20 [==============================] - 12s 612ms/step - loss: 0.9735 - accuracy: 0.7805 - va
l_loss: 1.3134 - val_accuracy: 0.6390
Epoch 91/150
20/20 [==============================] - 14s 685ms/step - loss: 0.9154 - accuracy: 0.8073 - va
l_loss: 1.2417 - val_accuracy: 0.6740
Epoch 92/150
20/20 [==============================] - 14s 689ms/step - loss: 0.9443 - accuracy: 0.7960 - va
l_loss: 1.2967 - val_accuracy: 0.6920
Epoch 93/150
20/20 [==============================] - 13s 625ms/step - loss: 0.9246 - accuracy: 0.8102 - va
l_loss: 1.3675 - val_accuracy: 0.6360
Epoch 94/150
20/20 [==============================] - 14s 675ms/step - loss: 0.9682 - accuracy: 0.7900 - va
l_loss: 1.3426 - val_accuracy: 0.6860
Epoch 95/150
20/20 [==============================] - 14s 694ms/step - loss: 0.9788 - accuracy: 0.7860 - va
l_loss: 1.2124 - val_accuracy: 0.7040
Epoch 96/150
20/20 [==============================] - 14s 702ms/step - loss: 0.8940 - accuracy: 0.8213 - va
l_loss: 1.3049 - val_accuracy: 0.6950
Epoch 97/150
20/20 [==============================] - 13s 653ms/step - loss: 0.9439 - accuracy: 0.8050 - va
l_loss: 1.3230 - val_accuracy: 0.6780
Epoch 98/150
20/20 [==============================] - 13s 639ms/step - loss: 0.9535 - accuracy: 0.8073 - va
l_loss: 1.4205 - val_accuracy: 0.6810
Epoch 99/150
20/20 [==============================] - 14s 688ms/step - loss: 1.0760 - accuracy: 0.7542 - va
l_loss: 1.2428 - val_accuracy: 0.6990
Epoch 100/150
20/20 [==============================] - 13s 668ms/step - loss: 0.9451 - accuracy: 0.8127 - va
l_loss: 1.2182 - val_accuracy: 0.7200
Epoch 101/150
20/20 [==============================] - 13s 637ms/step - loss: 0.9302 - accuracy: 0.8075 - va
l_loss: 1.2816 - val_accuracy: 0.7150
Epoch 102/150
20/20 [==============================] - 13s 673ms/step - loss: 0.9076 - accuracy: 0.8207 - va
l_loss: 1.1971 - val_accuracy: 0.7090
Epoch 103/150
20/20 [==============================] - 13s 644ms/step - loss: 0.8582 - accuracy: 0.8397 - va
l_loss: 1.2660 - val_accuracy: 0.6840
Epoch 104/150
20/20 [==============================] - 13s 648ms/step - loss: 0.9588 - accuracy: 0.7962 - va
l_loss: 1.3870 - val_accuracy: 0.6900
Epoch 105/150
20/20 [==============================] - 12s 608ms/step - loss: 0.9555 - accuracy: 0.8060 - va
l_loss: 1.3157 - val_accuracy: 0.7160
Epoch 106/150
20/20 [==============================] - 13s 649ms/step - loss: 0.9029 - accuracy: 0.8242 - va
l_loss: 1.3012 - val_accuracy: 0.7040
Epoch 107/150
20/20 [==============================] - 13s 634ms/step - loss: 0.8752 - accuracy: 0.8317 - va
l_loss: 1.2157 - val_accuracy: 0.7080
Epoch 108/150
20/20 [==============================] - 13s 673ms/step - loss: 0.8747 - accuracy: 0.8267 - va
l_loss: 1.2019 - val_accuracy: 0.7020
Epoch 109/150
20/20 [==============================] - 14s 704ms/step - loss: 0.8981 - accuracy: 0.8215 - va
l_loss: 1.2254 - val_accuracy: 0.7040
Epoch 110/150
20/20 [==============================] - 13s 666ms/step - loss: 0.9238 - accuracy: 0.8175 - va
l_loss: 1.1893 - val_accuracy: 0.7230
```

```
Epoch 111/150
20/20 [==============================] - 14s 713ms/step - loss: 0.9616 - accuracy: 0.7922 - va
l_loss: 1.3167 - val_accuracy: 0.6740
Epoch 112/150
20/20 [==============================] - 14s 686ms/step - loss: 0.9351 - accuracy: 0.8145 - va
l_loss: 1.3253 - val_accuracy: 0.6670
Epoch 113/150
20/20 [==============================] - 14s 687ms/step - loss: 0.9282 - accuracy: 0.8180 - va
l_loss: 1.4493 - val_accuracy: 0.6870
Epoch 114/150
20/20 [==============================] - 14s 703ms/step - loss: 0.9599 - accuracy: 0.7993 - va
l_loss: 1.2799 - val_accuracy: 0.6840
Epoch 115/150
20/20 [==============================] - 12s 608ms/step - loss: 0.8968 - accuracy: 0.8290 - va
l_loss: 1.8162 - val_accuracy: 0.5560
Epoch 116/150
20/20 [==============================] - 13s 651ms/step - loss: 1.1539 - accuracy: 0.7153 - va
l_loss: 1.6060 - val_accuracy: 0.5920
Epoch 117/150
20/20 [==============================] - 13s 636ms/step - loss: 1.0708 - accuracy: 0.7525 - va
l_loss: 1.2723 - val_accuracy: 0.7050
Epoch 118/150
20/20 [==============================] - 12s 622ms/step - loss: 0.9551 - accuracy: 0.7995 - va
l_loss: 1.2877 - val_accuracy: 0.6940
Epoch 119/150
20/20 [==============================] - 12s 612ms/step - loss: 0.8928 - accuracy: 0.8260 - va
l_loss: 1.2620 - val_accuracy: 0.6950
Epoch 120/150
20/20 [==============================] - 13s 643ms/step - loss: 0.8420 - accuracy: 0.8440 - va
l_loss: 1.2819 - val_accuracy: 0.7120
Epoch 121/150
20/20 [==============================] - 12s 616ms/step - loss: 0.8686 - accuracy: 0.8418 - va
l_loss: 1.6686 - val_accuracy: 0.6480
Epoch 122/150
20/20 [==============================] - 13s 663ms/step - loss: 0.9680 - accuracy: 0.7950 - va
l_loss: 1.1815 - val_accuracy: 0.7210
Epoch 123/150
20/20 [==============================] - 14s 683ms/step - loss: 0.9133 - accuracy: 0.8190 - va
l_loss: 1.3029 - val_accuracy: 0.7070
Epoch 124/150
20/20 [==============================] - 14s 690ms/step - loss: 0.8465 - accuracy: 0.8410 - va
l_loss: 1.3585 - val_accuracy: 0.7020
Epoch 125/150
20/20 [==============================] - 13s 629ms/step - loss: 0.8939 - accuracy: 0.8220 - va
l_loss: 1.2992 - val_accuracy: 0.6940
Epoch 126/150
20/20 [==============================] - 14s 677ms/step - loss: 0.8812 - accuracy: 0.8263 - va
l_loss: 1.2512 - val_accuracy: 0.6990
Epoch 127/150
20/20 [==============================] - 13s 657ms/step - loss: 0.8717 - accuracy: 0.8332 - va
l_loss: 1.3638 - val_accuracy: 0.6630
Epoch 128/150
20/20 [==============================] - 13s 619ms/step - loss: 0.8809 - accuracy: 0.8342 - va
l_loss: 1.2938 - val_accuracy: 0.7270
Epoch 129/150
20/20 [==============================] - 13s 662ms/step - loss: 0.8715 - accuracy: 0.8372 - va
l_loss: 1.3591 - val_accuracy: 0.6680
Epoch 130/150
20/20 [==============================] - 14s 701ms/step - loss: 0.9130 - accuracy: 0.8158 - va
l_loss: 1.4229 - val_accuracy: 0.6580
Epoch 131/150
20/20 [==============================] - 14s 698ms/step - loss: 1.0063 - accuracy: 0.7793 - va
l_loss: 1.5037 - val_accuracy: 0.5930
Epoch 132/150
20/20 [==============================] - 14s 685ms/step - loss: 0.9566 - accuracy: 0.8040 - va
l_loss: 1.2440 - val_accuracy: 0.7100
```

```
Epoch 133/150
20/20 [==============================] - 13s 652ms/step - loss: 0.9282 - accuracy: 0.8120 - va
l_loss: 1.1941 - val_accuracy: 0.7140
Epoch 134/150
20/20 [==============================] - 14s 679ms/step - loss: 0.8972 - accuracy: 0.8240 - va
l_loss: 1.2523 - val_accuracy: 0.6920
Epoch 135/150
20/20 [==============================] - 14s 675ms/step - loss: 0.9847 - accuracy: 0.7845 - va
l_loss: 1.2782 - val_accuracy: 0.6750
Epoch 136/150
20/20 [==============================] - 10s 506ms/step - loss: 0.9474 - accuracy: 0.8002 - va
l_loss: 1.5418 - val_accuracy: 0.6530
Epoch 137/150
20/20 [==============================] - 10s 510ms/step - loss: 0.9174 - accuracy: 0.8175 - va
l_loss: 1.2390 - val_accuracy: 0.6990
Epoch 138/150
20/20 [==============================] - 13s 644ms/step - loss: 0.9188 - accuracy: 0.8085 - va
l_loss: 1.4276 - val_accuracy: 0.6800
Epoch 139/150
20/20 [==============================] - 11s 554ms/step - loss: 0.9391 - accuracy: 0.8015 - va
l_loss: 1.3604 - val_accuracy: 0.6710
Epoch 140/150
20/20 [==============================] - 10s 496ms/step - loss: 0.9729 - accuracy: 0.7853 - va
l_loss: 1.3378 - val_accuracy: 0.6430
Epoch 141/150
20/20 [==============================] - 11s 532ms/step - loss: 0.8672 - accuracy: 0.8330 - va
l_loss: 1.3197 - val_accuracy: 0.7020
Epoch 142/150
20/20 [==============================] - 10s 484ms/step - loss: 0.8242 - accuracy: 0.8612 - va
l_loss: 1.3184 - val_accuracy: 0.6890
Epoch 143/150
20/20 [==============================] - 10s 517ms/step - loss: 0.8760 - accuracy: 0.8250 - va
l_loss: 1.1989 - val_accuracy: 0.7230
Epoch 144/150
20/20 [==============================] - 10s 492ms/step - loss: 0.8683 - accuracy: 0.8320 - va
l_loss: 1.3153 - val_accuracy: 0.7060
Epoch 145/150
20/20 [==============================] - 10s 516ms/step - loss: 0.8242 - accuracy: 0.8540 - va
l_loss: 1.2756 - val_accuracy: 0.7120
Epoch 146/150
20/20 [==============================] - 10s 511ms/step - loss: 0.8015 - accuracy: 0.8658 - va
l_loss: 1.3503 - val_accuracy: 0.6850
Epoch 147/150
20/20 [==============================] - 10s 512ms/step - loss: 0.9056 - accuracy: 0.8160 - va
l_loss: 1.2880 - val_accuracy: 0.6740
Epoch 148/150
20/20 [==============================] - 9s 478ms/step - loss: 0.9892 - accuracy: 0.7810 - val
_loss: 1.2629 - val_accuracy: 0.6710
Epoch 149/150
20/20 [==============================] - 10s 525ms/step - loss: 0.9194 - accuracy: 0.8130 - va
l_loss: 1.3253 - val_accuracy: 0.6880
Epoch 150/150
20/20 [==============================] - 10s 487ms/step - loss: 0.8555 - accuracy: 0.8495 - va
l_loss: 1.3136 - val_accuracy: 0.6960
Accuracy:  0.8495000004768372
 Validation accuracy:  0.6959999799728394
```

Training and validation plots

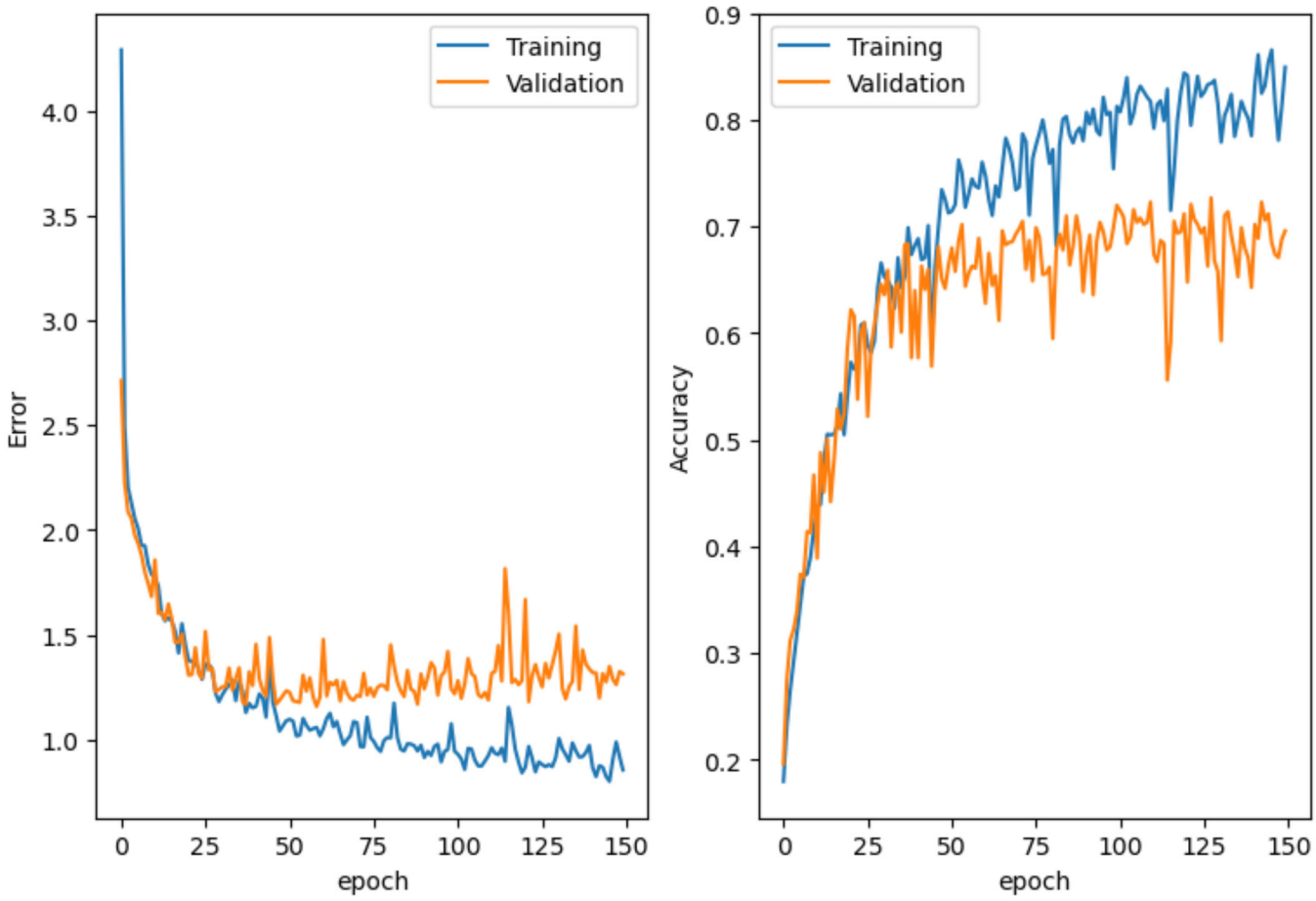
In [8]:

```
plt.figure(1, figsize = (9,6))
plt.subplot(121)
# Plot the training error
plt.plot(estimator.history['loss'], label = 'Training')
plt.plot(estimator.history['val_loss'], label = 'Validation')
plt.ylabel('Error')
```

```
plt.xlabel('epoch')
plt.legend()
#plt.show()

# Plot the accuracy
plt.subplot(122)
plt.plot(estimator.history['accuracy'], label = 'Training')
plt.plot(estimator.history['val_accuracy'], label = 'Validation')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x1a68907b1f0>



Evaluate

Time to evaluate the model with the test data

In [10]:
```
print('Evaluating model: ')
scores = model.evaluate(xtest, ytest, batch_size = 128) #prints out score.
print("test loss, test acc:", round(scores[0], 4), round(scores[1], 4))
```

```
Evaluating model:
40/40 [==============================] - 3s 75ms/step - loss: 1.3130 - accuracy: 0.7074
test loss, test acc: 1.313 0.7074
```

Results summary

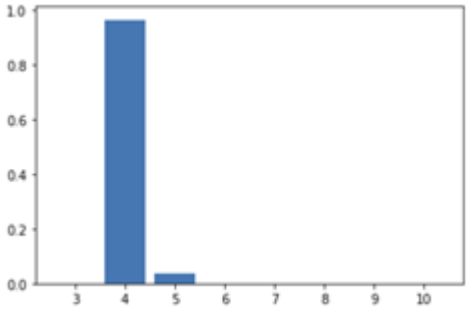
- Around 70% polygons were correctly classified
- Model is slightly overtrained

What went wrong? Examples of correct and wrongly classified polygons

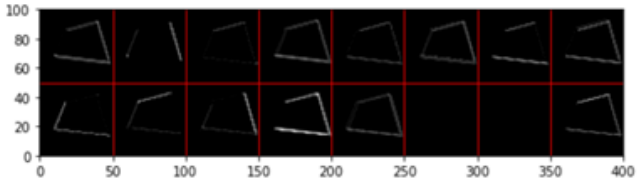
- Showing typical example of polygons that were harder to identify
- Display the filters

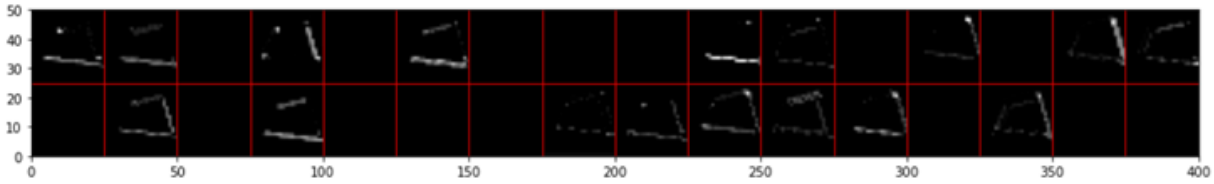
Starting with a correctly classified polygon:

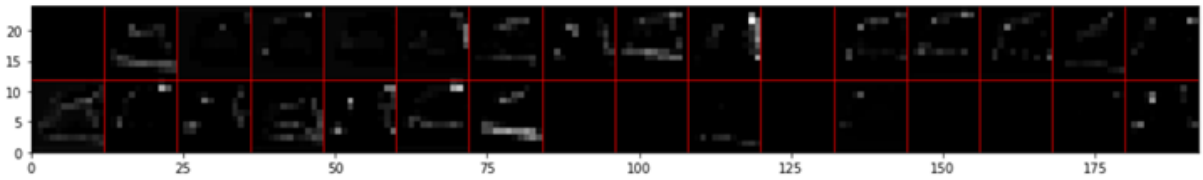
predicted as 4 corners:

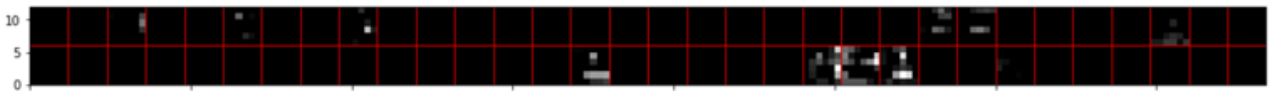


the filters convoluted with this polygon look like this:





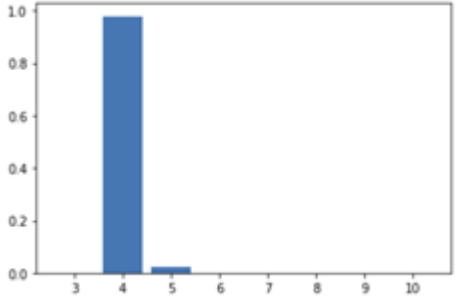




- Note how the edges are identified in the earlier convolutional filters

and an example of a wrongly classified polygon:

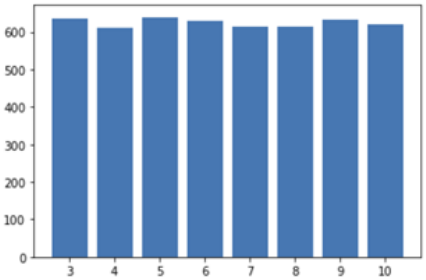
classified as:



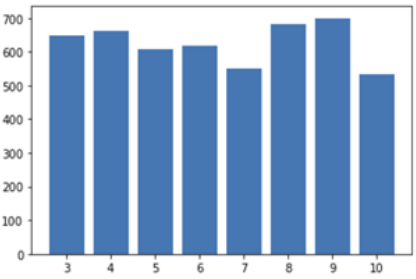
Note that:

- one of the corners are close to 180 degrees which is most probably the corner that is not counted
- there is an existing, but lower, propability for 5 corners

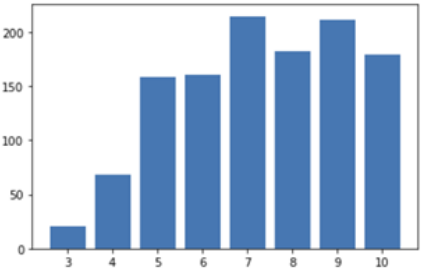
Statistics



Ground truth



Prediction



Incorrect prediction