Home Work Assignment 1

Or Kozlovsky 305447476

Noa Englender 307875302

Task 1: Deep Learning Introduction

Dry section

1. Working with a convolution network:
   a. the output dimensions of the following layers are with an Input RGB image of size 128X128X3:

Layer #	Layer description	output dimensions
1	convolution with 64 kernels of size 1X1X3	128X128X64
2	max pooling of size 2x2	64X64X64
3	convolution with 32 kernels of size 5X5X64 (no zero padding).	60X60X64

   b. We will explain the calculation of a 2D convolution with a kernel of size 1X1X3. The output of the convolution is the sum of the same pixel in all 3 channels of the image. i.e. let's assume we have the following kernel:

1	1	1

   And the following image channels:

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

   Then the output, as described above, will be:

3	6	9
12	15	18
21	24	27

   c. We will present the output of the convolution of the given sub-image with the following average filter:

$$\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$$

i. With stride = (1,1) and padding = (1,1):

$$\begin{matrix} 1.1111 & 2.5556 & 2.6667 & 2.8889 & 1.4444 \\ 1.8889 & 4.1111 & 4.3333 & 4.7778 & 2.5556 \\ 1.8889 & 4.2222 & 4.7778 & 4.8889 & 2.5556 \\ 1.4444 & 3.6667 & 4.2222 & 4.6667 & 2.4444 \\ 0.6667 & 2.1111 & 2.5556 & 2.7778 & 1.3333 \end{matrix}$$

ii. With stride = (2,1) and padding = (0,0):

$$\begin{matrix} 4.1111 & 4.7778 \\ 4.2222 & 4.8889 \\ 3.6667 & 4.6667 \\ 2.1111 & 2.7778 \end{matrix}$$

2.

Convolution layer - The convolution layer consistes of filters which are convolved with the input of the layer.
The kernels are a the set of learnable filters that make up the layer. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume and computes dot products between the entries of the filter and the input at any position. This produces a 2-dimensional activation map that gives the responses of that filter at every spatial position.
The parameters are the weights- the values of the filters.

The hyperparameters are:

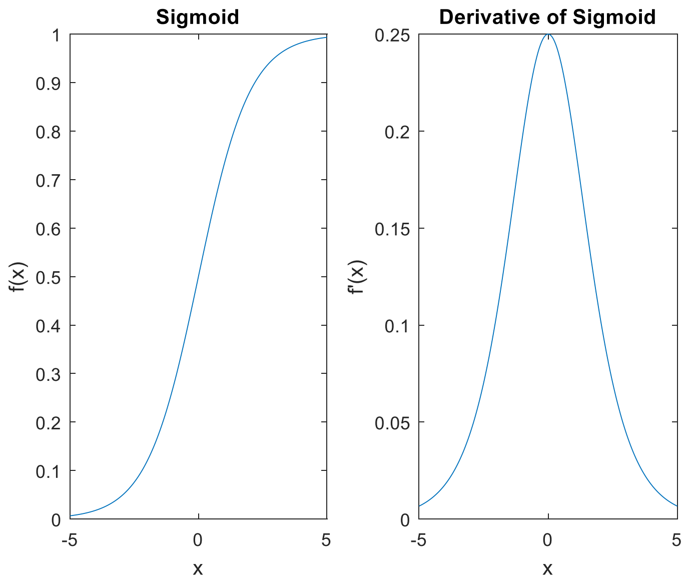
- filter size: affects the filter response
- depth of output of the layer: this is controlled by the number of filters used in the layer
- zero-padding: allows us to control the spatial size of the output
- Stride: A bigger stride in the convolution produces smaller output volumes spatially. (in width and height)

Pooling layer- This layer downsamples the input to the layer. Reducing the spatial size of the representation reduces the amount of parameters and computation in the network, and helps control overfitting. Pooling is done independently on every depth slice of the input and resizes it spatially. It Introduces zero parameters since it computes a fixed function of the input
Max pooling takes that maximum of the defined neighborhod (most commonly a 2x2 neighborhood) as the output.

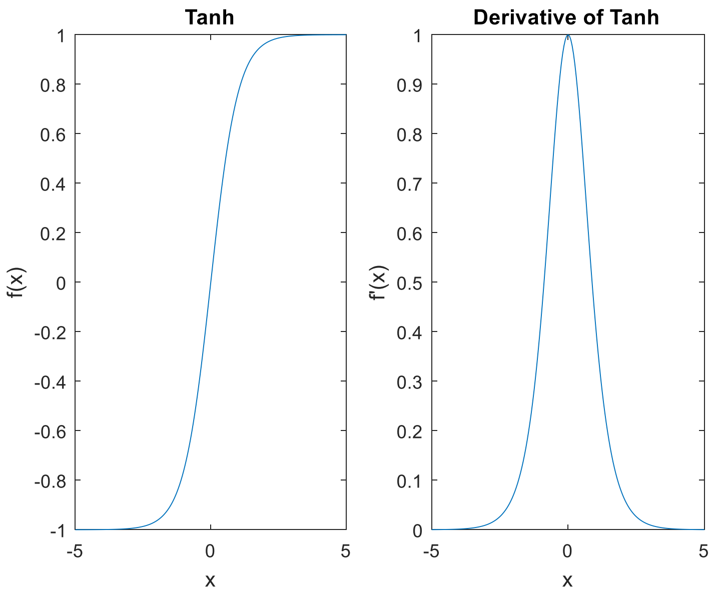
Average pooling takes the average of the neighborhood.
The hyperparameters for this layer are the spatial extent and the stride of the operation.

Non-Linear Activation Functions-

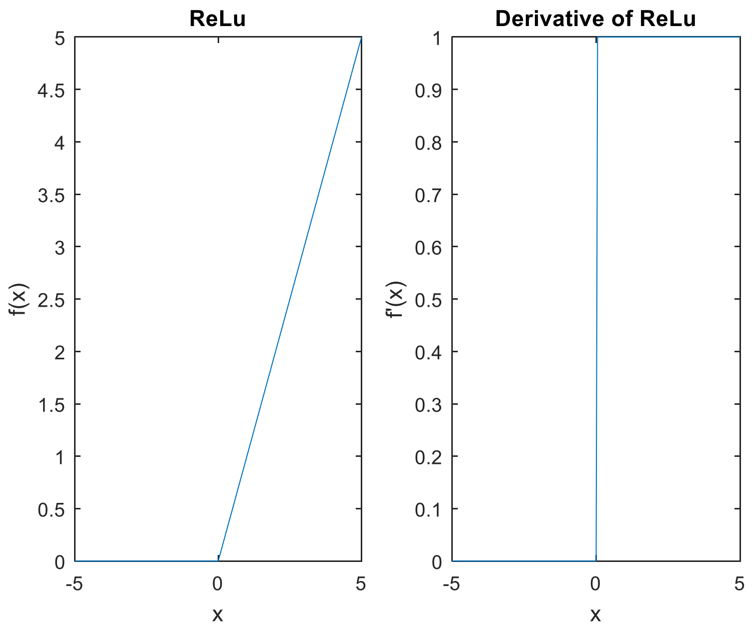
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}},\ f'(x) = f(x) \cdot (1 - f(x))$



- Thanh (hyperbolic tangent): $f(x) = \frac{(1-\mathrm{e}^{-2x})}{(1-\mathrm{e}^{-2x})}\ , f'(x) = 1 - f^2(x)$



- ReLu: $f(x) = \max(0, x)\,, f'(x) = \begin{cases} 1\,, & x > 0 \\ 0, & else \end{cases}$



3. Image Classification problem is the task of assigning an input image one label from a fixed set of categories.
The ImageNet project is a large visual database. Over 14 million URLs of images have been hand-annotated by ImageNet to indicate what objects are pictured. ImageNet contains over 20 thousand ambiguous categories. The ImageNet challenge is a competition where research teams evaluate their algorithms on the given data set, and compete to achieve higher accuracy on several visual recognition tasks.
The definition of Image Classification in ImageNet is: “For each image, algorithms will produce a list of at most 5 object categories in the descending order of confidence. The quality of a labeling will be evaluated based on the label that best matches the ground truth label for the image.”

4. We selcted to present the VGG16 classification architecture:

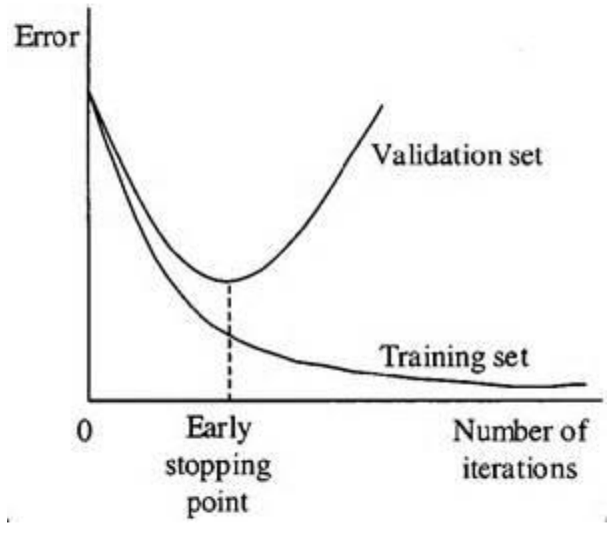
Layer #	Layer description	output dimensions	Kernel size	Stride (dx, dy)	Padding\dilation (Dx, Dy)
0	input	224x224x3	-	-	-
1	Convolution	224x224X64	3X3	(1, 1)	(1, 1)
2	Convolution	224x224X64	3X3	(1, 1)	(1, 1)
3	max pooling	112x112x64	2X2	(2, 2)	(1, 1)
4	Convolution	112x112x128	3X3	(1, 1)	(1, 1)
5	Convolution	112x112x128	3X3	(1, 1)	(1, 1)
6	max pooling	56X56x128	2X2	(2, 2)	(1, 1)
7	Convolution	56X56x256	3X3	(1, 1)	(1, 1)
8	Convolution	56X56x256	3X3	(1, 1)	(1, 1)
9	Convolution	56X56x256	3X3	(1, 1)	(1, 1)
10	max pooling	28X28x256	2X2	(2, 2)	(1, 1)
11	Convolution	28X28x512	3X3	(1, 1)	(1, 1)
12	Convolution	28X28x512	3X3	(1, 1)	(1, 1)

13	Convolution	28X28x512	3X3	(1, 1)	(1, 1)
14	max pooling	14X14x512	2X2	(2, 2)	(1, 1)
15	Convolution	14X14x512	3X3	(1, 1)	(1, 1)
16	Convolution	14X14x512	3X3	(1, 1)	(1, 1)
17	Convolution	14X14x512	3X3	(1, 1)	(1, 1)
18	max pooling	7X7x512	2X2	(2, 2)	(1, 1)
19	Flatten	1x25088	-	-	-
20	Fully connected	1x4096	-	-	-
21	Fully connected	1x4096	-	-	-
22	Prediction (out)	1x1000	-	-	-

- The output of one layer is the input of the sequential layer.
- The input size, kernel size, stride and padding dictate the output size of each layer as described in proviso sections

5. overfitting is when our trained model (net) doesn't generalize well from our training data to unseen data. When a model learns the noise (training set) instead of the signal (observed (true) values) is considered "overfit" because it fits the training dataset but has poor fit with new datasets. We can recognize it for example, if our model saw 99% accuracy on the training set but only 55% accuracy on the validation set.

   We can measure how well each iteration of the model performs when training a learning algorithm iteratively. Up until a certain number of iterations, new iterations improve the model accuracy. After that point, however, the model's ability to generalize can weaken as it begins to overfit the training data.
   Early stopping refers stopping the training process before the learner passes that point.



In the above graph we can recognize two cases:

1. The error (accuracy) of the Training set which decrease as monotonously as the number of iterations.

2. The error of the Validation set which from a certain point begins to increase as the iterations continue.

   Our goal when building our learning algorithm is to recognize this point and make sure we not passing it.

6. During training, we aim to minimize the loss function. This is done by updating the network parameters by taking small steps in the direction of the negative gradient of the loss function. This is called “backpropagation”:
   Forward pass:

   1. Forward through the network to generate the predicted output values.
   2. Calculation of the cost (loss function)

   Back propagation:

   Starting with the gradient of the loss function, back propagate calculations of gradients in order to update the weights of each layer, so that the with each iteration the gradient of the loss function is minimized.

7. To train a NN, normally two datasets are used:

1. One dataset is the training set ("gold standard"), it contains the input data together with correct/expected output.
2. The other dataset is the validation/testing set. It contains data to apply the model to in order to test its accuracy.

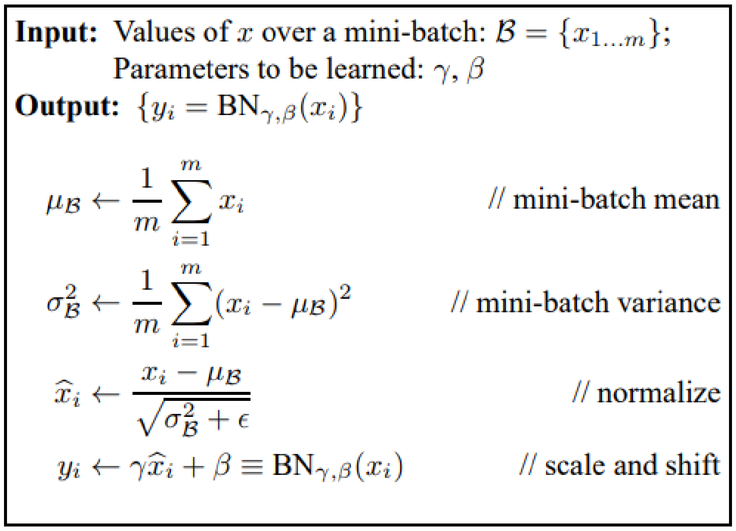
Training phase: you present your data from your "gold standard" and train your model, by pairing the input with expected output. The loss function is calculated for every output and then backpropagation is calculated to adjust the weights.

Validation/Test phase: in order to estimate how well the model has been trained and to estimate model properties, the test set is run through the network. This phase does not update the weights; thus, it involves less computation. Therefore, **test takes less time to run than training.**

8. Batch normalization layer introduces some kind of regularization to the network and helps the convergence procedure.
   The empirical mean and std of a layer's activations are calculated, and normalized accordingly.
   Each BN layer has two learned parameters that are updated with the weights during backprop.

$$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}.$$

   If the optimal thing to do is keep the activations as they were, the learned parameters will converge back to the mean and variance.
   The following algorithms describes the normalization process:



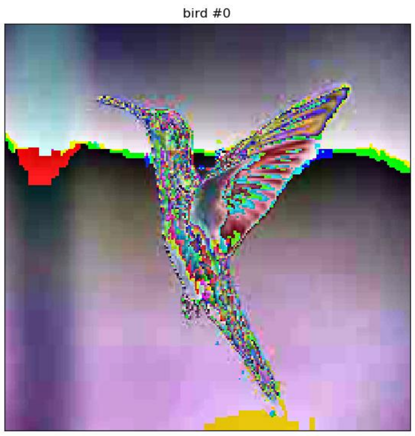
```
Input: Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
       Parameters to be learned: $\gamma$, $\beta$
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

    $\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                          // mini-batch mean
    $\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2$          // mini-batch variance
    $\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}$                     // normalize
    $y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$            // scale and shift
```

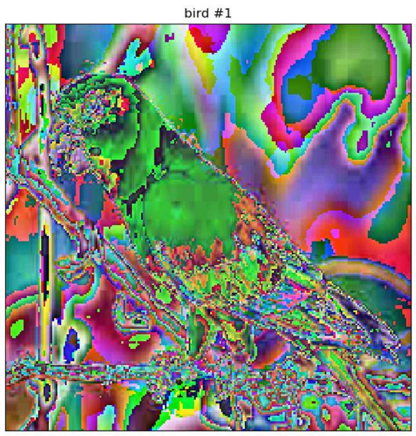
Wet section

2.

The Original birds image

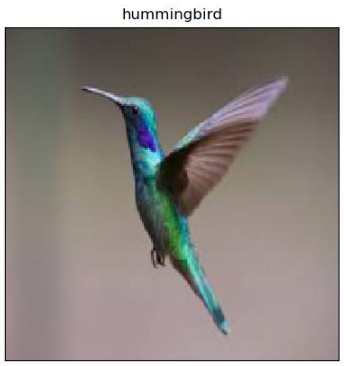
The transformed image – fitting as an input to VGG16

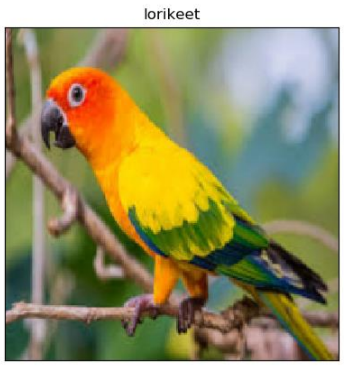




The output of the network is a vector in which each index contains the energy for the corresponding class. The index containing the highest value is the index of the most likely classification.
The model succeeded in classifying the images:

The model (VGG16) output

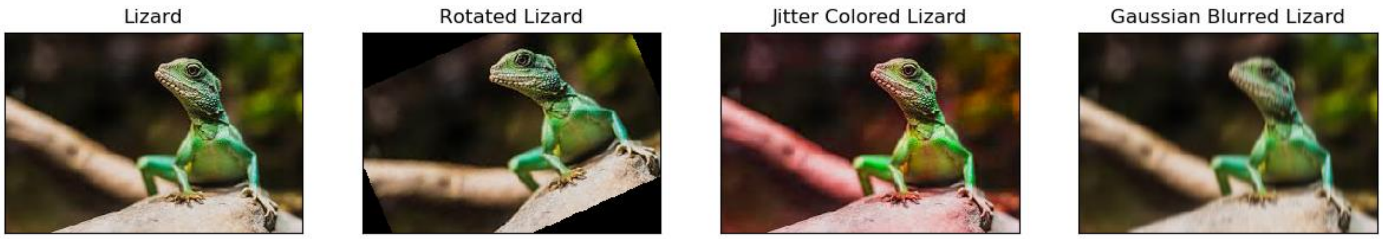




3.

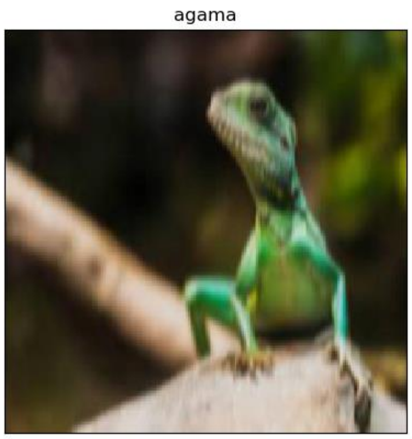
<u>The model (VGG16) output</u>

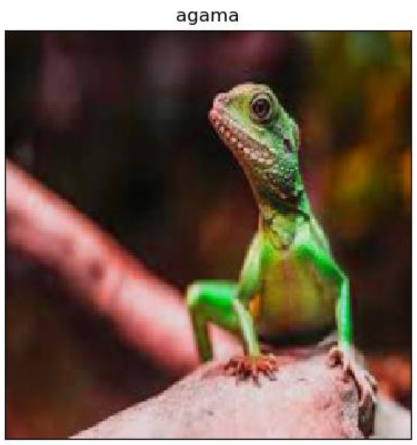
4. **<u>The original image and the 3 transformations:</u>**



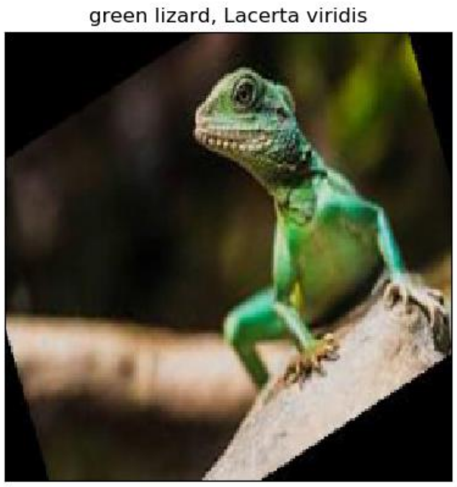
<u>The network outputs for the transformed images:</u>

Output for the gaussian blurred image was classified the same as the original image. So was the jitter color transformed image:



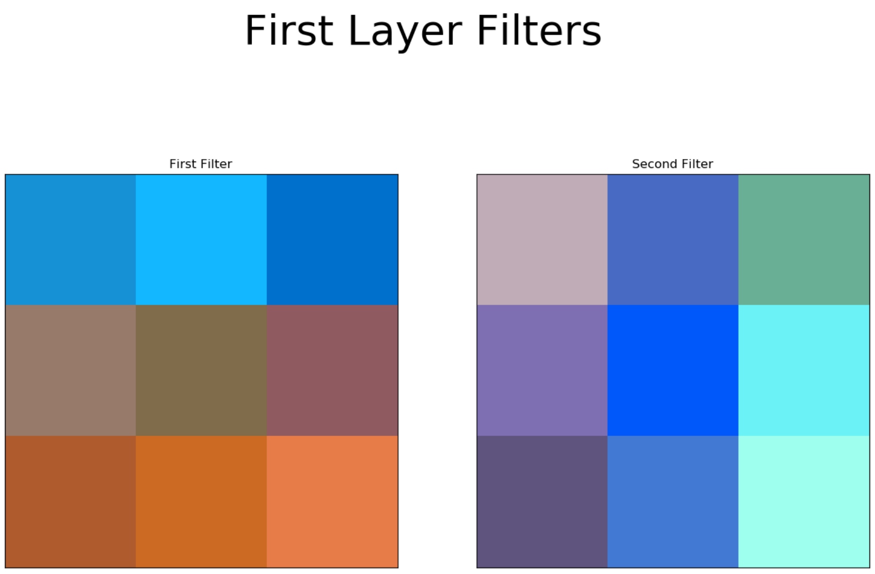


However, the rotated image was not classified correctly:

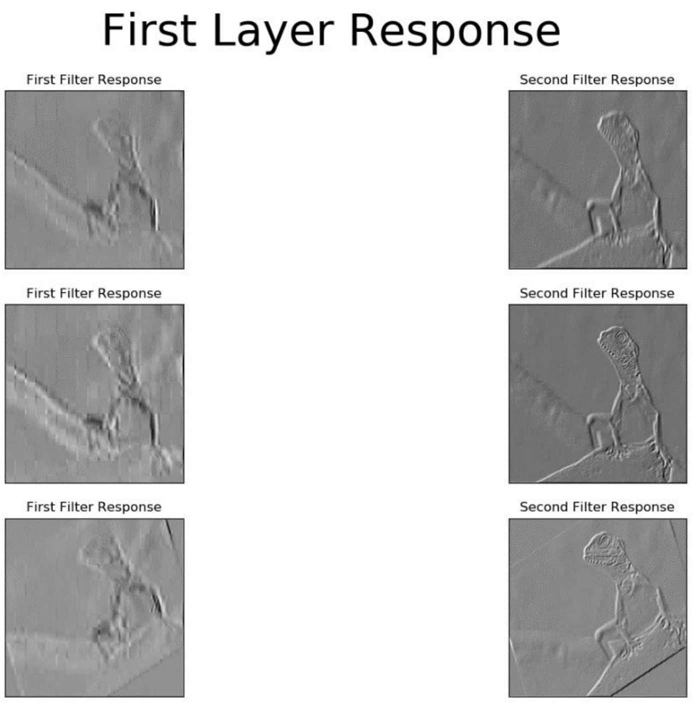


5.

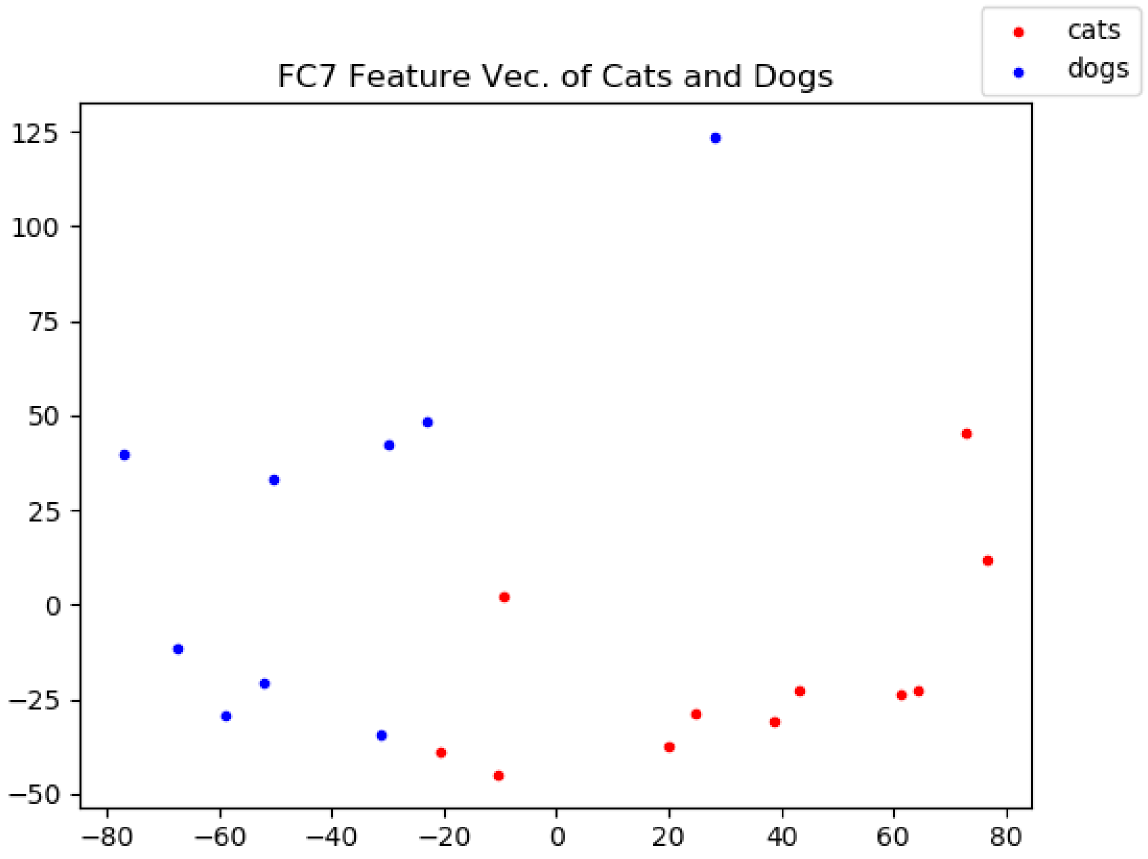
<u>2 filters from the first conv. Layer of VGG16:</u>



<u>The filter responses to the transformed images:</u>

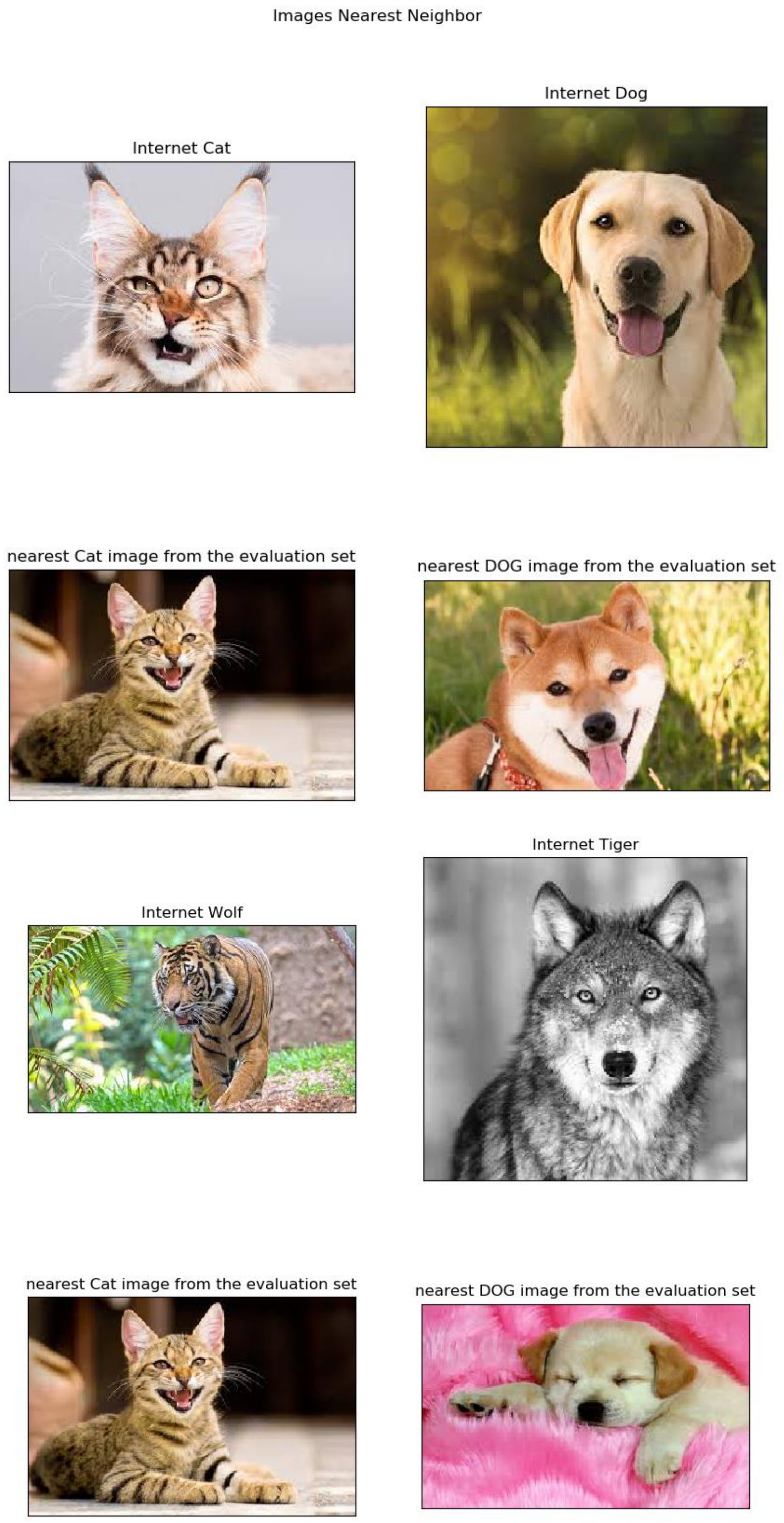


6. We extracted the feature vector of layer FC7 for the cats and dogs images. Since the output of the layer is a vector of size 4096, plotting all these vectors on one graph does not give a good visual representation of the different features.
   In order to better visualize the most important features, we used PCA on the vectors and displayed the 2 primary components:



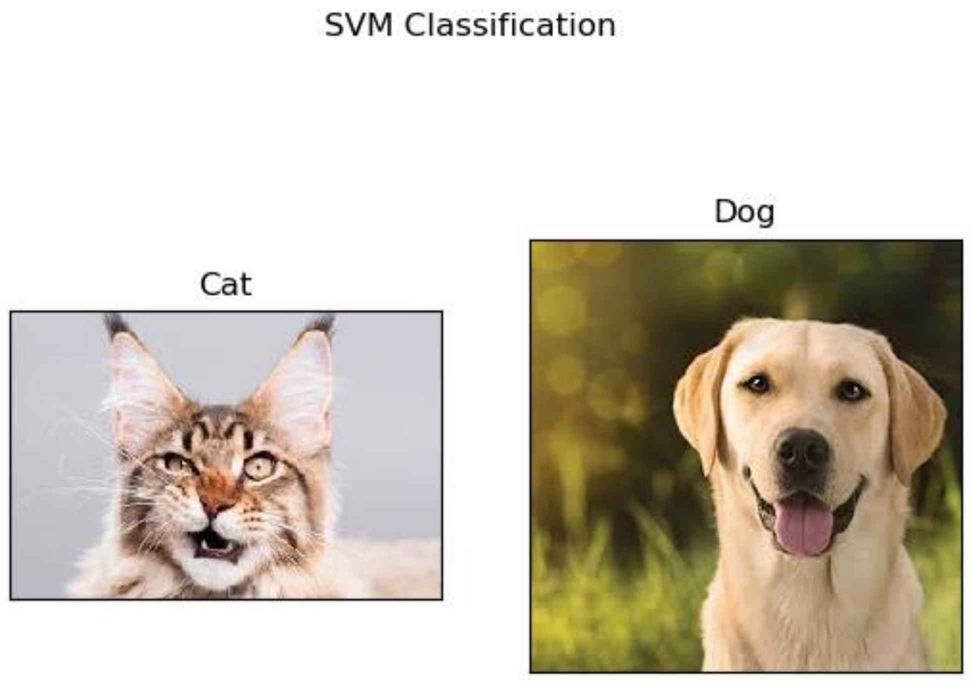
It is possible to draw a line to separate the cats vectors from the dogs vectors, meaning this layer already extracted important features from the images, which allow to distinguish between the two classes.

7. + 8. In order to calculate the nearest neighbor, we used the Euclidean distance.


Images Nearest Neighbor
Internet Cat
Internet Dog
nearest Cat image from the evaluation set
nearest DOG image from the evaluation set
Internet Wolf
Internet Tiger
nearest Cat image from the evaluation set
nearest DOG image from the evaluation set

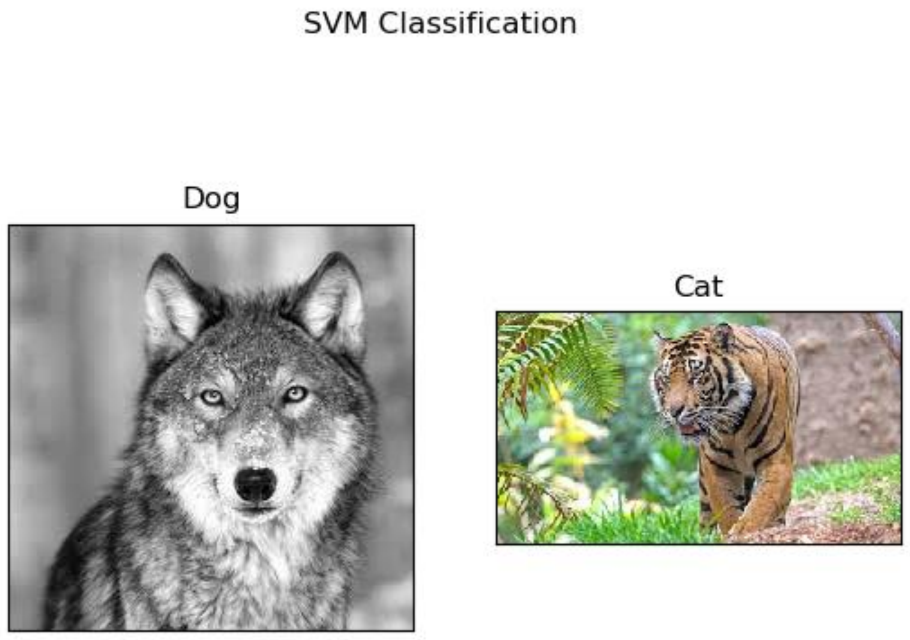
9. Like we saw section 6, the output features vector of the FC7 layer can be used to distinguish between vectors from the cats or the dogs classes. Thus, we chose to use the outputs of the FC7 layer as inputs to the SVM classifier.

SVM classification for an image of a cat and an image of a dog from the internet:



10. We ran an image of a tiger and an image of a wolf through the classifier trained to classify cats and dogs:

    SVM classification for an image of a tiger and an image of a wolf from the internet:



Task 2: Edge detection

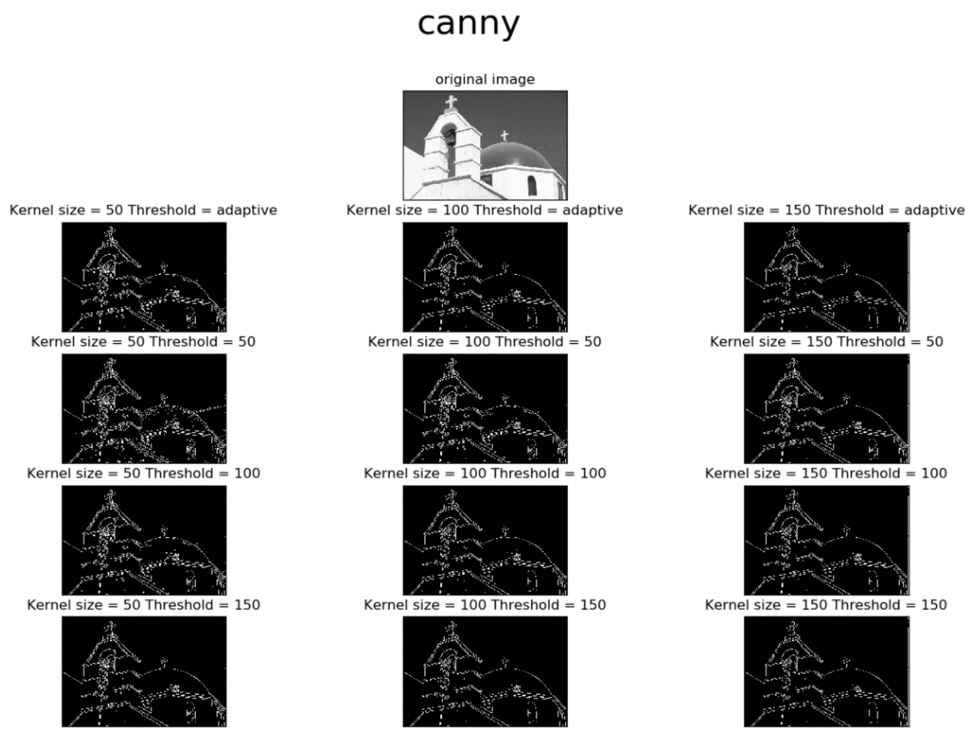
A.

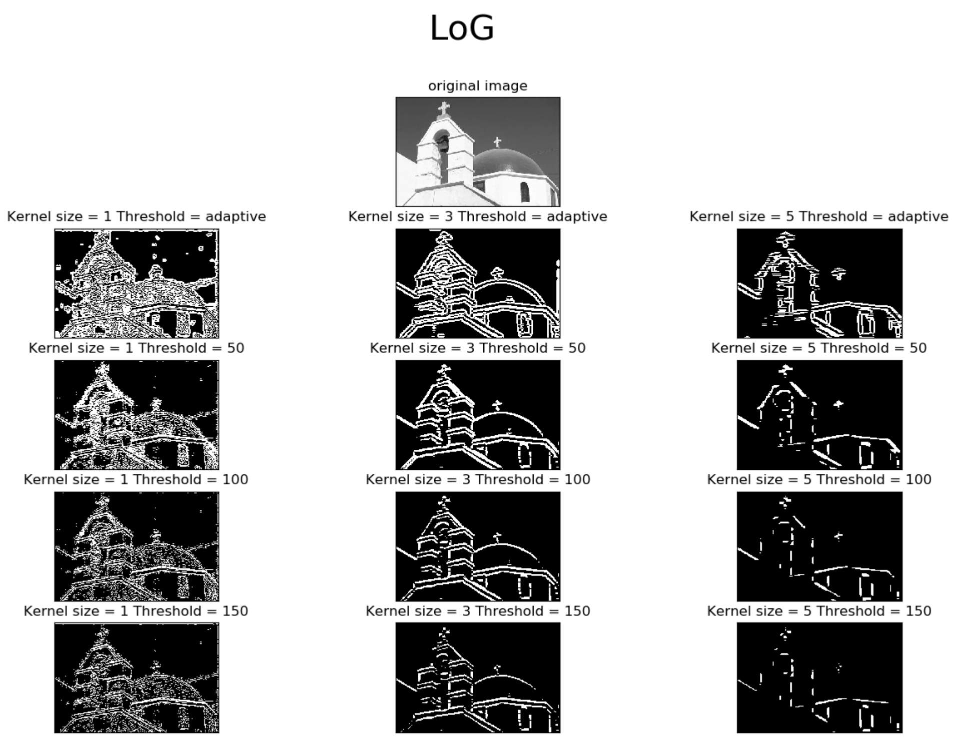
1. We will explain the way of operation of each of the three detectors:
    - Gaussian-Laplace: highlights regions of rapid intensity change and is therefore used for edge detection. It calculates the Laplacian of the image given by the relation, out $= \frac{\partial^2 src}{\partial x} + \frac{\partial^2 src}{\partial y}$. the operator uses a 3x3 kernel $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ which convolved with the original image to calculate approximations of the derivatives. to suppress the noise before using the Laplacian we use a Gaussian kernel.
    - Sobel: works like the first order derivate and calculates the difference of pixel intensities. The center column is of zero, so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. Also, the center values of both the first and third column is 2 and -2 respectively.
      This give more weight to the pixel values around the edge region. This increase the edge intensity and it became enhanced comparatively to the original image.
    - Canny: Canny edge detection is a multi-step algorithm that can detect edges with noise suppressed at the same time:
        i. Smooth the image with a Gaussian filter.
        ii. Compute the gradient.
        iii. Appling a threshold.
        iv. Suppress non-maxima pixels to thin the edge ridges.
        v. Threshold the previous result by two different thresholds.
        vi. Link edge segments in the two imaged from the proviso step.

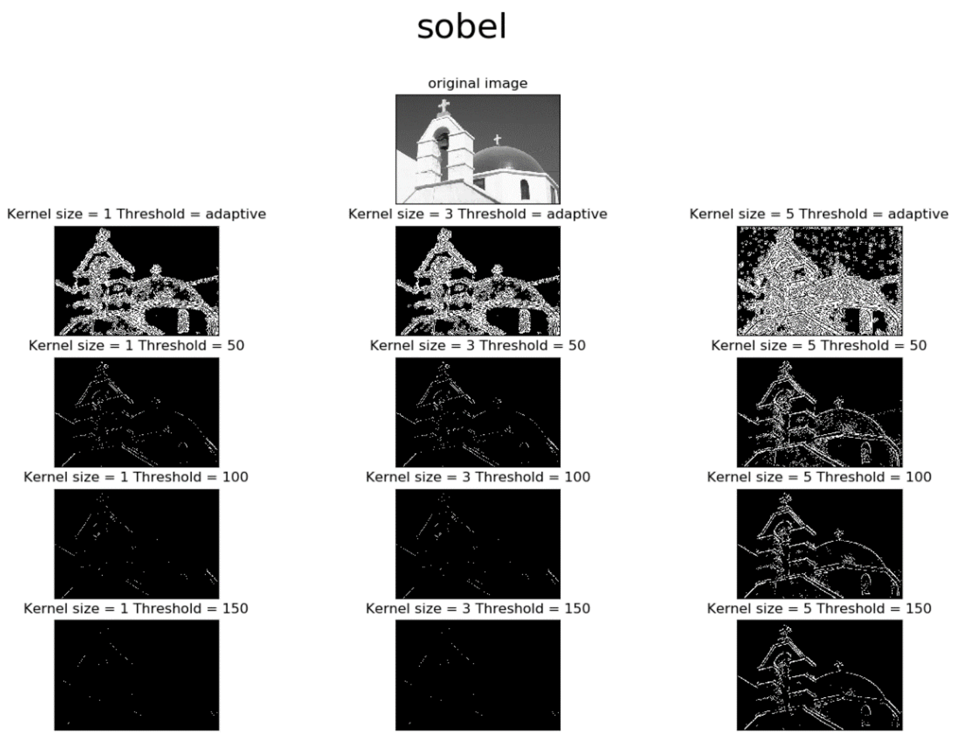
2. We will explain the threshold parameter and the sigma parameter of the detectors:

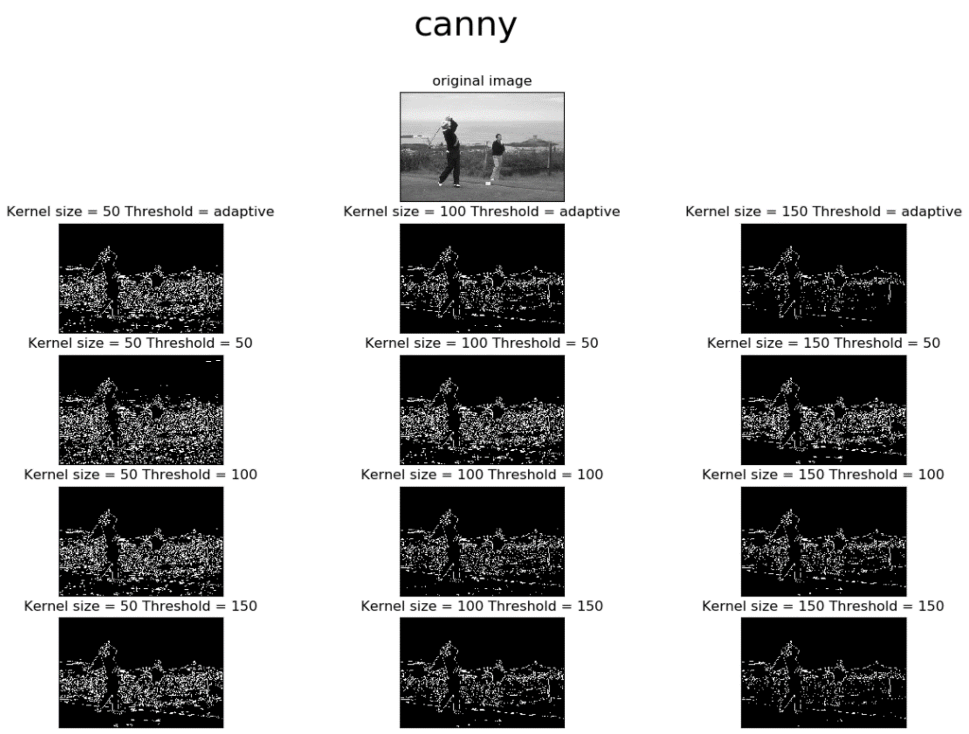
	threshold	sigma
Canny	Edges with intensity gradient more than max threshold are sure to be edges and those below min threshold are sure to be non-edges, so discarded. Those that lie between the two thresholds are classified edges or non-edges based on their connectivity. If they are connected to “sure-edge” pixels, they are part of edges. Otherwise, they are also discarded. Increasing the top thres. Can cause real edges to be missed while decreasing it can cause false edge detections.	The standard deviations of the Gaussian filter - determines the smoothing factor. When the sigma decreases then we give more weight to the pixels closer to the centered pixel, thus reducing the smoothing effect.
Gaussian-Laplace	all values above the threshold determined as edges (1) and all below as non-edges (0) Increasing the thres. Can cause real edges to be missed while decreasing it can cause false edge detections.	
Sobel		-

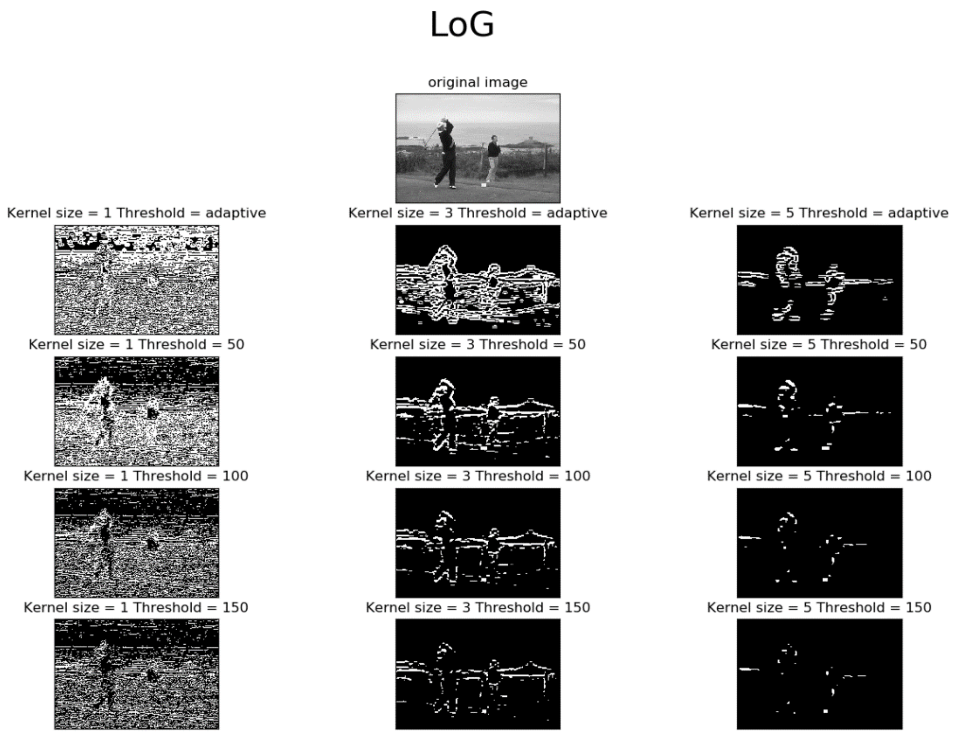
Below are the results for using the 3 methods of edge detection on the 3 given images, using different parameters and thresholds:


canny
original image
Kernel size = 50 Threshold = adaptive
Kernel size = 100 Threshold = adaptive
Kernel size = 150 Threshold = adaptive
Kernel size = 50 Threshold = 50
Kernel size = 100 Threshold = 50
Kernel size = 150 Threshold = 50
Kernel size = 50 Threshold = 100
Kernel size = 100 Threshold = 100
Kernel size = 150 Threshold = 100
Kernel size = 50 Threshold = 150
Kernel size = 100 Threshold = 150
Kernel size = 150 Threshold = 150

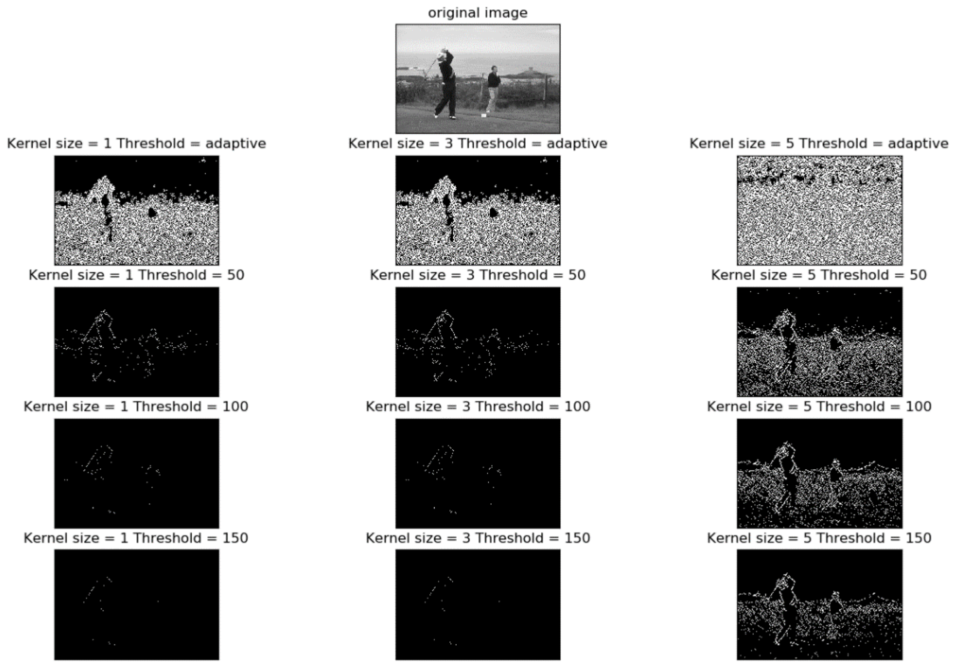

LoG
original image
Kernel size = 1 Threshold = adaptive
Kernel size = 3 Threshold = adaptive
Kernel size = 5 Threshold = adaptive
Kernel size = 1 Threshold = 50
Kernel size = 3 Threshold = 50
Kernel size = 5 Threshold = 50
Kernel size = 1 Threshold = 100
Kernel size = 3 Threshold = 100
Kernel size = 5 Threshold = 100
Kernel size = 1 Threshold = 150
Kernel size = 3 Threshold = 150
Kernel size = 5 Threshold = 150


sobel
original image
Kernel size = 1 Threshold = adaptive
Kernel size = 3 Threshold = adaptive
Kernel size = 5 Threshold = adaptive
Kernel size = 1 Threshold = 50
Kernel size = 3 Threshold = 50
Kernel size = 5 Threshold = 50
Kernel size = 1 Threshold = 100
Kernel size = 3 Threshold = 100
Kernel size = 5 Threshold = 100
Kernel size = 1 Threshold = 150
Kernel size = 3 Threshold = 150
Kernel size = 5 Threshold = 150

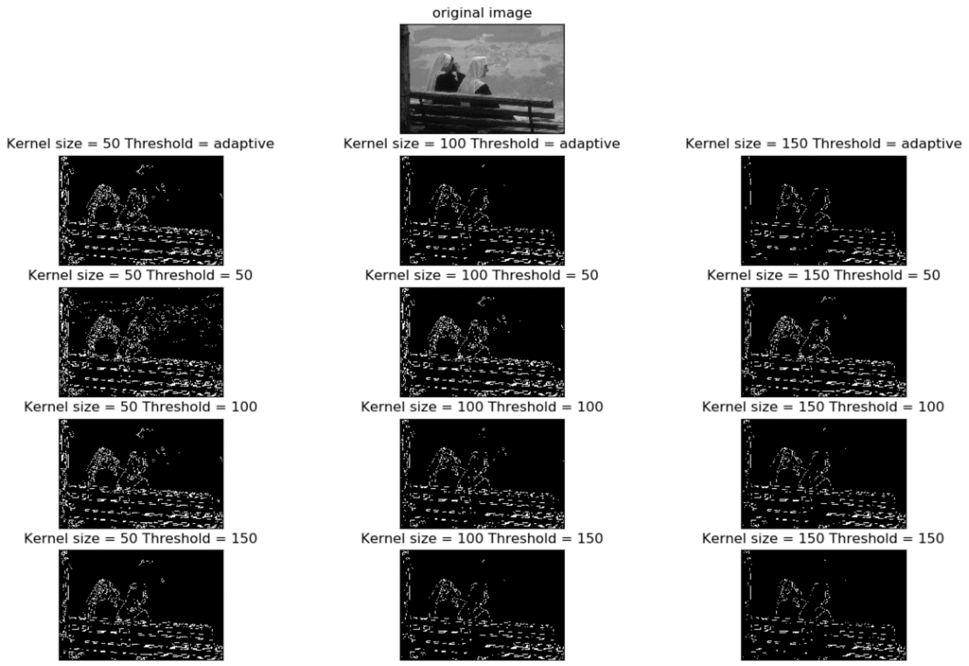

canny
original image
Kernel size = 50 Threshold = adaptive
Kernel size = 100 Threshold = adaptive
Kernel size = 150 Threshold = adaptive
Kernel size = 50 Threshold = 50
Kernel size = 100 Threshold = 50
Kernel size = 150 Threshold = 50
Kernel size = 50 Threshold = 100
Kernel size = 100 Threshold = 100
Kernel size = 150 Threshold = 100
Kernel size = 50 Threshold = 150
Kernel size = 100 Threshold = 150
Kernel size = 150 Threshold = 150


LoG
original image
Kernel size = 1 Threshold = adaptive
Kernel size = 3 Threshold = adaptive
Kernel size = 5 Threshold = adaptive
Kernel size = 1 Threshold = 50
Kernel size = 3 Threshold = 50
Kernel size = 5 Threshold = 50
Kernel size = 1 Threshold = 100
Kernel size = 3 Threshold = 100
Kernel size = 5 Threshold = 100
Kernel size = 1 Threshold = 150
Kernel size = 3 Threshold = 150
Kernel size = 5 Threshold = 150

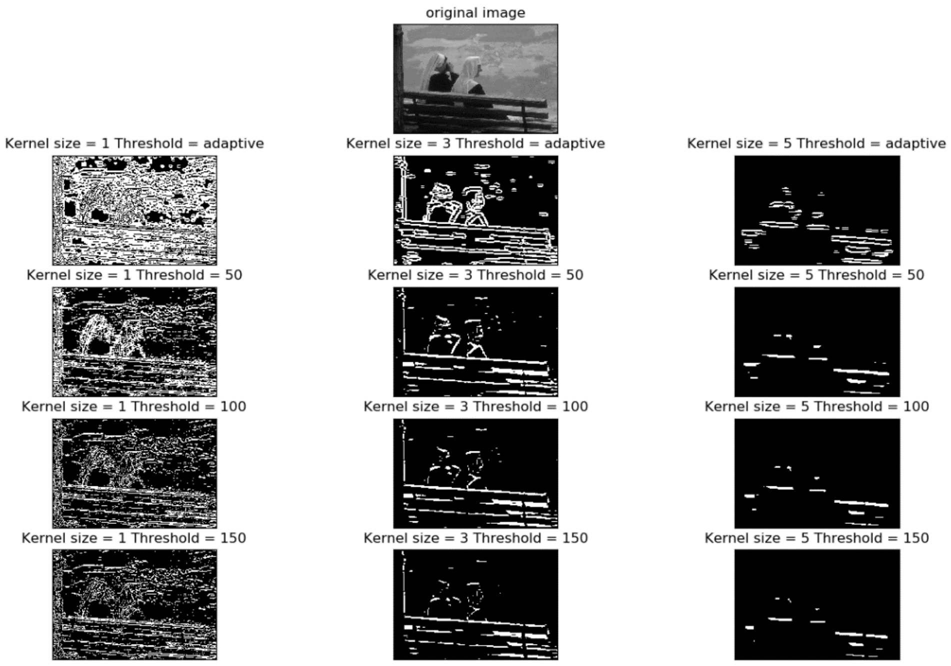
sobel


original image
Kernel size = 1 Threshold = adaptive
Kernel size = 3 Threshold = adaptive
Kernel size = 5 Threshold = adaptive
Kernel size = 1 Threshold = 50
Kernel size = 3 Threshold = 50
Kernel size = 5 Threshold = 50
Kernel size = 1 Threshold = 100
Kernel size = 3 Threshold = 100
Kernel size = 5 Threshold = 100
Kernel size = 1 Threshold = 150
Kernel size = 3 Threshold = 150
Kernel size = 5 Threshold = 150

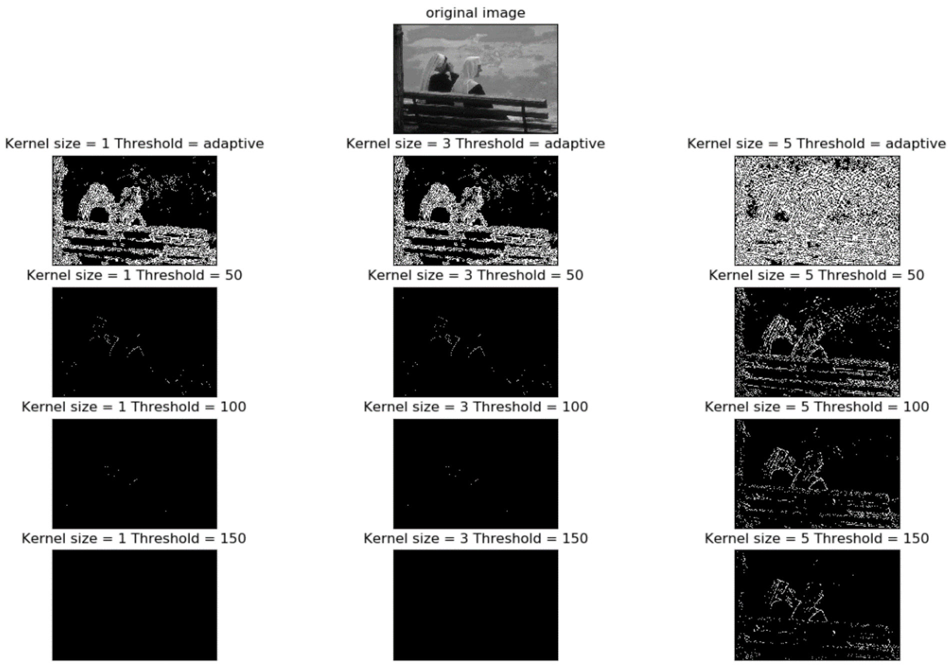
canny


original image
Kernel size = 50 Threshold = adaptive
Kernel size = 100 Threshold = adaptive
Kernel size = 150 Threshold = adaptive
Kernel size = 50 Threshold = 50
Kernel size = 100 Threshold = 50
Kernel size = 150 Threshold = 50
Kernel size = 50 Threshold = 100
Kernel size = 100 Threshold = 100
Kernel size = 150 Threshold = 100
Kernel size = 50 Threshold = 150
Kernel size = 100 Threshold = 150
Kernel size = 150 Threshold = 150

LoG


original image
Kernel size = 1 Threshold = adaptive
Kernel size = 3 Threshold = adaptive
Kernel size = 5 Threshold = adaptive
Kernel size = 1 Threshold = 50
Kernel size = 3 Threshold = 50
Kernel size = 5 Threshold = 50
Kernel size = 1 Threshold = 100
Kernel size = 3 Threshold = 100
Kernel size = 5 Threshold = 100
Kernel size = 1 Threshold = 150
Kernel size = 3 Threshold = 150
Kernel size = 5 Threshold = 150

sobel


original image
Kernel size = 1 Threshold = adaptive
Kernel size = 3 Threshold = adaptive
Kernel size = 5 Threshold = adaptive
Kernel size = 1 Threshold = 50
Kernel size = 3 Threshold = 50
Kernel size = 5 Threshold = 50
Kernel size = 1 Threshold = 100
Kernel size = 3 Threshold = 100
Kernel size = 5 Threshold = 100
Kernel size = 1 Threshold = 150
Kernel size = 3 Threshold = 150
Kernel size = 5 Threshold = 150

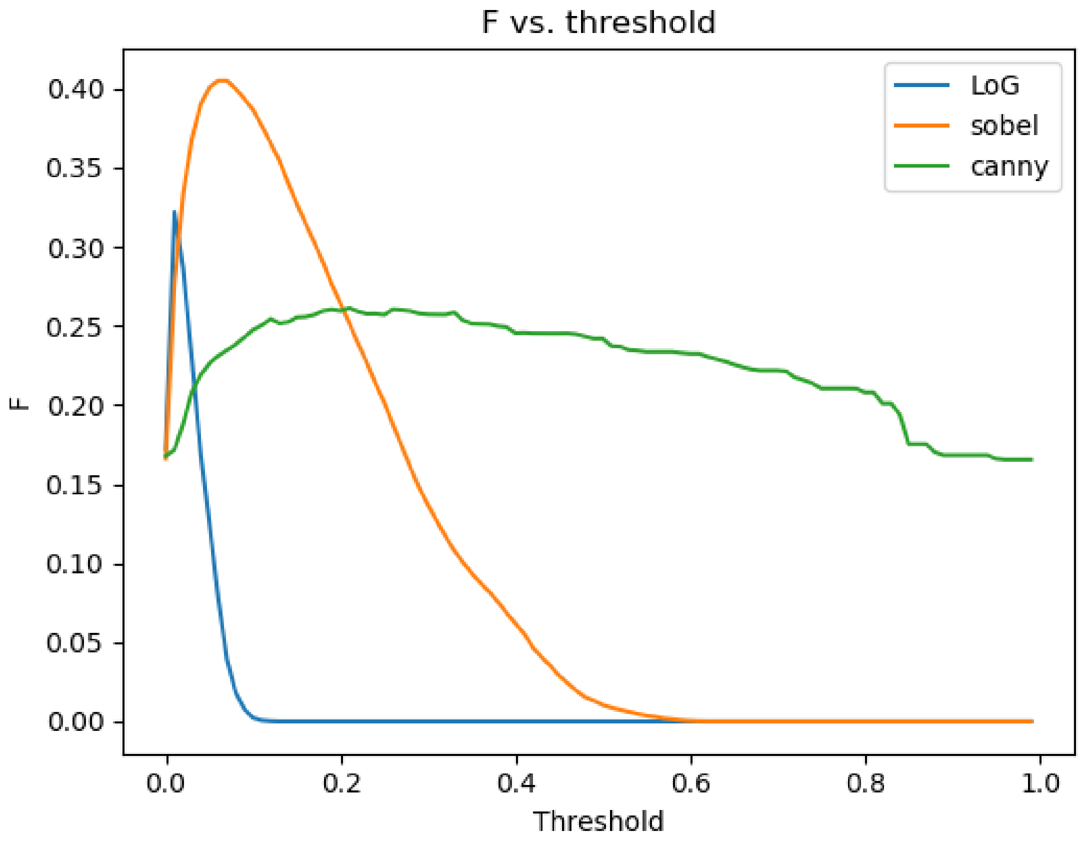
Overall, it looks like Canny performs best with kernel size of 100 and threshold of 100 or 150. LoG performs best with kernel size of 3 and threshold of 50, and sobel best with kernel suze if 5 and threshold of 150.

(in this part the pixel values were between 0 and 255 so the thresholds were chosen accordingly. In the next section we scaled the images and the thresholds to be between 0 and 1)

B.

1. A precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C. However, precision says nothing about the number of items from class C that were not labeled correctly.
   A recall of 1.0 means that every item from class C was labeled as belonging to class C, but recall says nothing about how many other items were incorrectly also labeled as belonging to class C.
   We want higher precision for tasks where correctly classifying is important (for example diagnosing a tumor from tomography images- we wouldn’t want to miss a tumor)
   And we want higher recall when It is important to not mis-classify (for example: classifying which part of the brain is a tumor in order to remove the cells. We wouldn’t want to remove healthy cells.)

2.


F vs. threshold
LoG
sobel
canny
F
Threshold
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.0
0.2
0.4
0.6
0.8
1.0