046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

אושר להגשה באיחור על ידי תמר

Dry Section

1. Working with a convolution network
   a. Write the output dimensions of every layer for a network with 3 layers:
      - Input: RGB image of size 128X128X3
      - Layer 1: convolution with 64 kernels of size 1X1X3: **128X128 x 64 images**
      - Layer 2: max pooling of size 2x2 (stride 1): **127X127 x 64 images**
      - Layer 3: convolution with 32 kernels of size 5X5X64: **123X123 x 32 images**

   b. Explain the calculation of a 2D convolution with a kernel of size 1X1X(?) (a simple example will suffice).
   This convolution weights every value across the third dimension and sums it for every cell in the first two dimensions of the matrix. For example:
   A is an RGB of ones and b is a vector: $b = [0.1 \quad 0.2 \quad 0.3]$ so $C = A * b$ is a 2D matrix with only the value of 0.6
   c. Convolve the following sub-image with a normalized 3X3 filter of your choice (select and design a filter). Write which filter you chose. What is the convolution output? Show the results for two options of stride and padding.

$$I[m,n] = \begin{bmatrix} 4 & 1 & 6 & 1 & 3 \\ 3 & 2 & 7 & 7 & 2 \\ 2 & 5 & 7 & 3 & 7 \\ 1 & 4 & 7 & 1 & 3 \\ 0 & 1 & 6 & 4 & 4 \end{bmatrix}, \quad filter\ h[m,n] = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}$$

padding=0, stride = 1: [2, 0.5, 1; 1, 0, -3; -2, -0.5, 0.5]
padding =1, stride = 1:
[1.5, 1, 3.5, 3.5, 1; -1, 2, 0.5, 1, 2; -1, 1, 0, -3, 0.5; -1, -2, -0.5, 0.5, -1.5; -0.5, -2, -3.5, -0.5, -1.5]

padding=0, stride = 2: [2, 1; -2, 0.5]
padding=1, stride = 2: [1.5, 3.5, 1; -1, 0, -3; -0.5, -3.5, -1.5].

046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

2. Select an architecture suggested for image classification (e.g., one of VGG16, VGG19, AlexNet) and write the input and output sizes of each layer. If you make assumptions, e.g., of the stride and padding, please explain. We recommend you will search online for details- there are many resources.

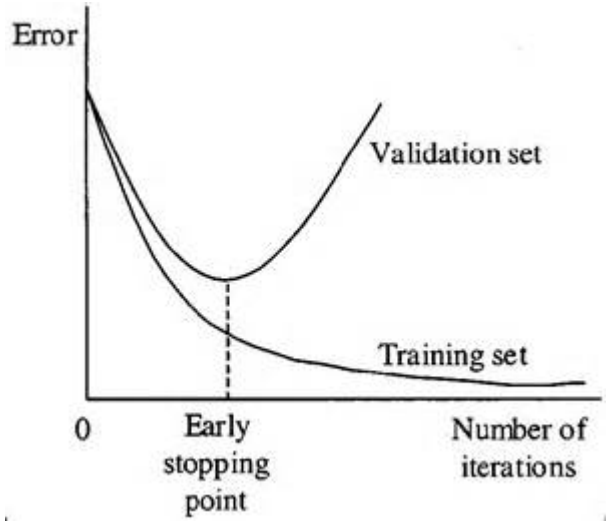
((**Alex Net**))

- **Layer 1: Input Layer: 227X227X3**
- **Layer 2: Conv 11X11, stride=4, 96 kernels: 55X55X96**
- Layer 3: ReLu (doesn't alter input/output dimensions)
- Layer 4: Batch-normalization (doesn't alter input/output dimensions)
- **Layer 5: MaxPool 3X3, stride=2: 27X27X96**
- **Layer 6: Conv 5X5, pad=2, stride=4, 256 kernels: 27X27X256**
- Layer 7: ReLu (doesn't alter input/output dimensions)
- Layer 8: Batch-normalization (doesn't alter input/output dimensions)
- **Layer 9: MaxPool 3X3, stride=2: 13X13X256**
- **Layer 10: Conv 3X3, pad=1, stride=4, 384 kernels: 13X13X384**
- Layer 11: ReLu (doesn't alter input/output dimensions)
- **Layer 12: Conv 3X3, pad=1, stride=4, 384 kernels: 13X13X384**
- Layer 13: ReLu (doesn't alter input/output dimensions)
- **Layer 14: Conv 3X3, pad=1, stride=4, 256 kernels: 13X13X256**
- Layer 15: ReLu (doesn't alter input/output dimensions)
- **Layer 16: MaxPool 3X3, stride=2: 6X6X256**
- **Layer 17: Fully Connected**: **4096X1**
- Layer 18: ReLu (doesn't alter input/output dimensions)
- Layer 19: Dropout 50% (doesn't alter input/output dimensions)
- **Layer 20: Fully Connected**: **4096X1**
- Layer 21: ReLu (doesn't alter input/output dimensions)
- Layer 22: Dropout (doesn't alter input/output dimensions)
- **Layer 23: Fully connected 1000X1**
- **Layer 24: SoftMax Classifier**: **1000**
- Layer 25: Cross-entropy loss (doesn't alter input/output dimensions)

3. What is overfitting and how can it be recognized during training? Draw a schematic accuracy vs. iterations graph. Discuss the different cases.

Overfitting is demonstrated in the following graph: The network's accuracy on the training set is expected to improve as more iterations are performed. The same should apply for accuracy on the validation set, up to a certain point.

This point usually marks the beginning of overfitting, which occurs when the network is over trained – becoming overfit for the training set, and thus isn't generalized to classify other sets.



046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

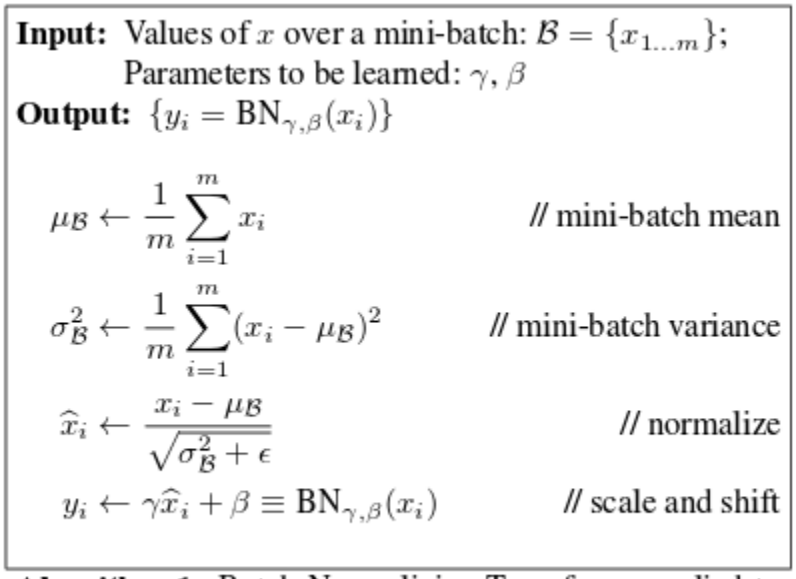
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

4. How do the learned parameters get updated during training?

   The network learns the parameters by minimizing a loss function. This is achieved using stochastic gradient descent, usually SGD (or other optimization method such as Momentum, Adaboost, Adagrad etc.). Calculating the gradient for each parameter is performed using the Backpropagation algorithm.

5. Explain the term batch normalization. Write down the formulation and give the definition for each variable.

   Batch normalization — processing of the data: a per-pixel mean μ_B and standard deviation σ_B^2 are calculated across all images of the batch. Then, each pixel is zerocentered and normalized to a standard deviation of 1. The result: Each pixel has a mean of 0 and a standard deviation of 1. Optionally, the network learns two additional parameters γ and β for optimal mean and standard deviation.



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

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

$$x_i = data\ inputs, \quad \widehat{x_i} = normazlied\ data, \quad \mu_B = average,$$
$$\sigma_B^2 = standard\ deviation \quad , \epsilon\ is\ a\ regularization\ paramater$$

046746 Computer Vision – HW2

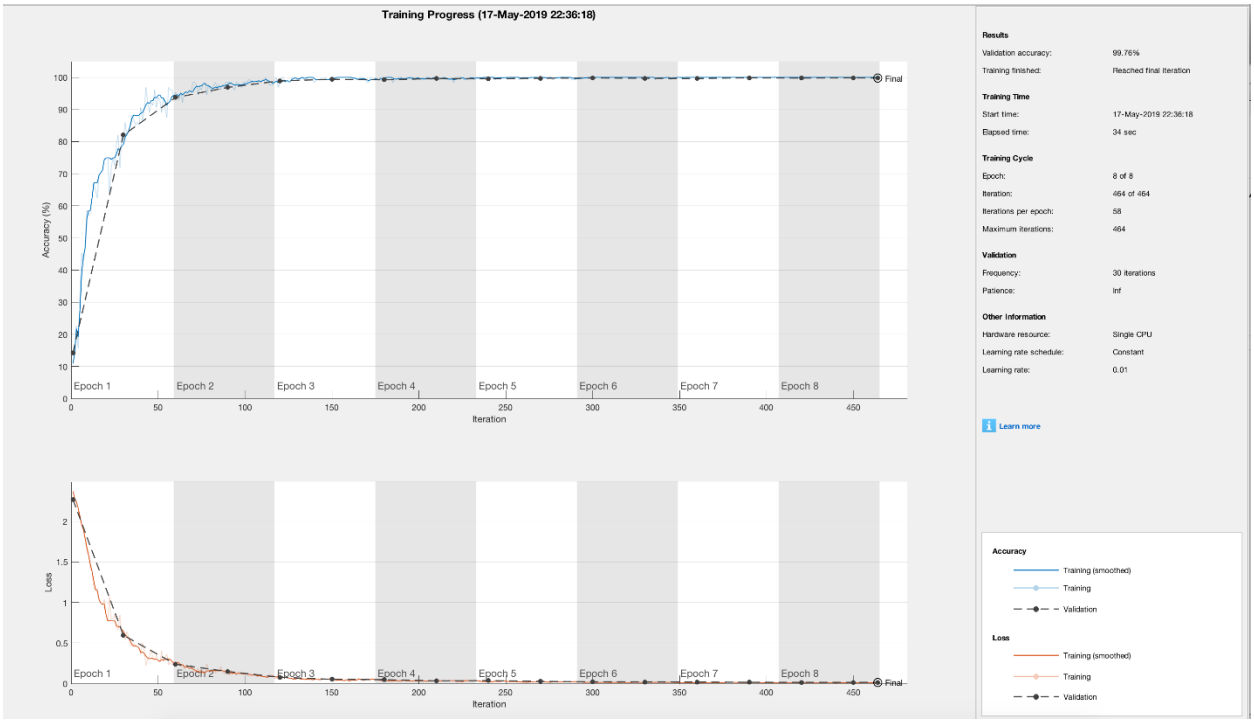
David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Task A – Training a simple CNN

Section 1-6:

Implemented and training network on MNIST dataset (split into 3:1 train-test ratio)
Learning rate **0.01**
Accuracy on test set: 99.72%



The network demonstrates a monotone increase in accuracy, reaching near maximal performance at Epoch2. A monotone reduction of the loss over the test-set implies that the network is learning efficiently.

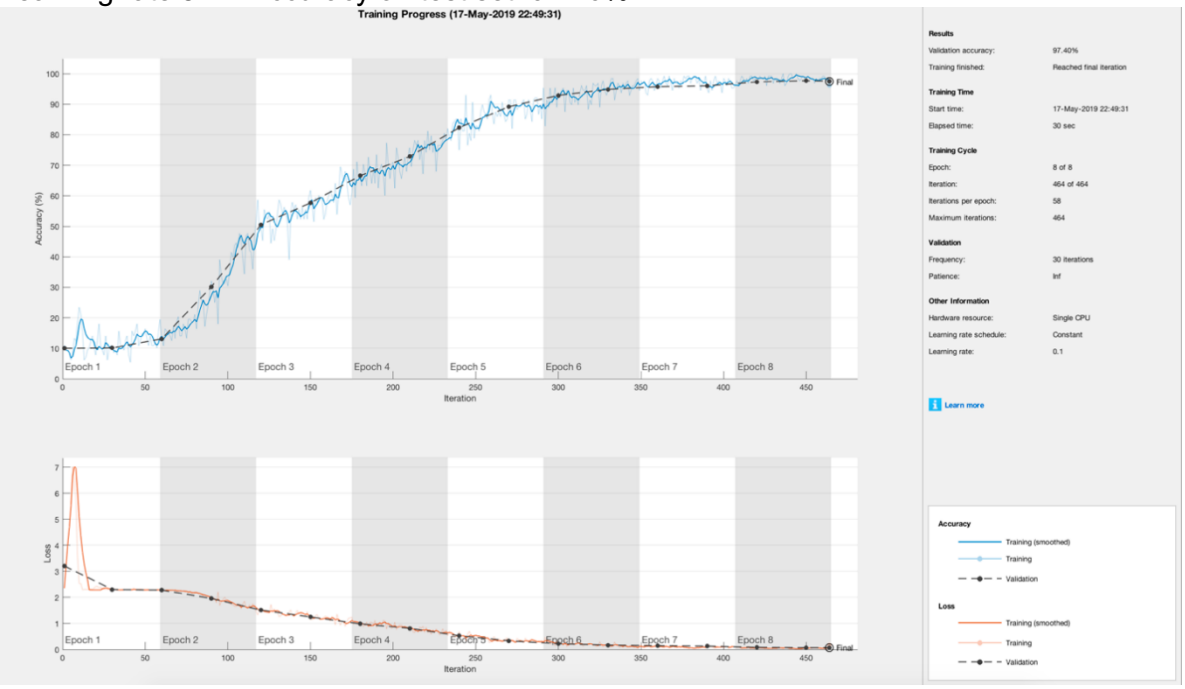
046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

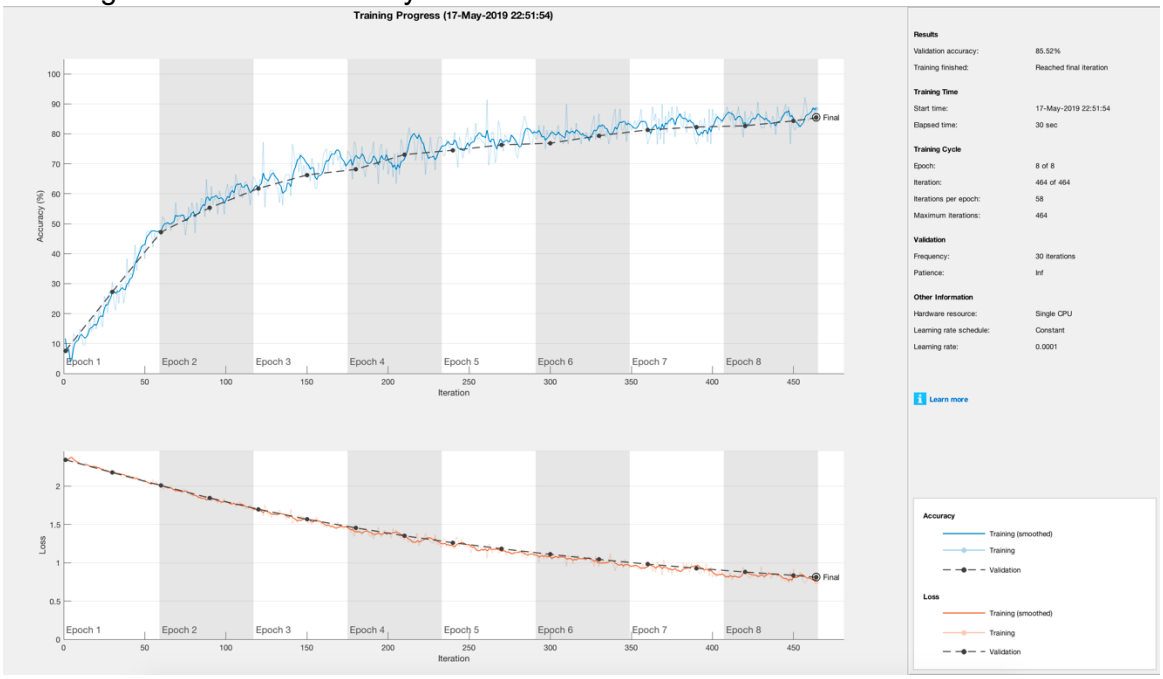
Section 7.1 – High Learning Rate

Learning rate **0.1** – Accuracy on test set: 97.40%



Section 7.2 – Small Learning Rate

Learning rate **0.0001** – Accuracy on test set: 85.52%



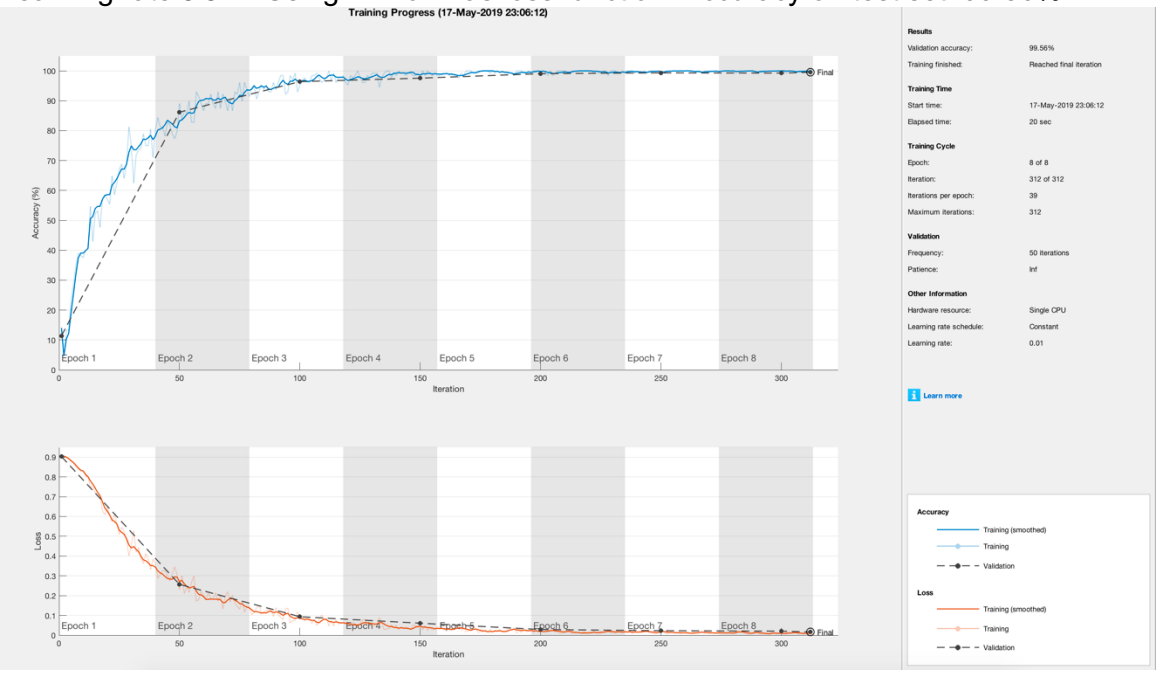
046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Section 7.3 – L2 loss

Learning rate **0.01** – Using L2 norm as loss function. Accuracy on test set: 99.56%



Section 7 – Conclusions

LR 0.01: For the cross-entropy loss and learning rate **0.01**, the network demonstrates a consistent monotone increase in accuracy and a consistent monotone reduction of the loss as a function of the iterations. We note that the network's classification accuracy exceeds 90% as soon as Epoch2 and nearly approaches its maximal value by Epoch4.

LR 0.1: In contrast, the network's performance deteriorates with a learning rate of **0.1**, as the rate is too high – resulting in a zig-zag of the loss during Epoch1, followed by an insufficient reduction of the loss (compared to a learning rate of **0.01**). The accuracy steadily increases; however, it only arrives the maximal value after Epoch8 (as opposed to Epoch4 with a learning rate of **0.01**) and falls short of the accuracy obtained in section 6.

LR 0.0001: Next, setting the learning rate to **0.0001** also produces inferior results: the learning rate is too small – evident by the monotonous but slow reduction of the loss, and with a corresponding slow increase in accuracy. The network does not reach near-maximal accuracy within the allotted epochs. This rate led to the lowest performance (85.52% vs +97%).

L2 loss, LR 0.01: The performance of the L2 loss are good but fall short of the performance achieved by the cross-entropy loss. Since the same network architecture is used, we can conclude that the cross-entropy is a loss function that is more appropriate for the classification task.

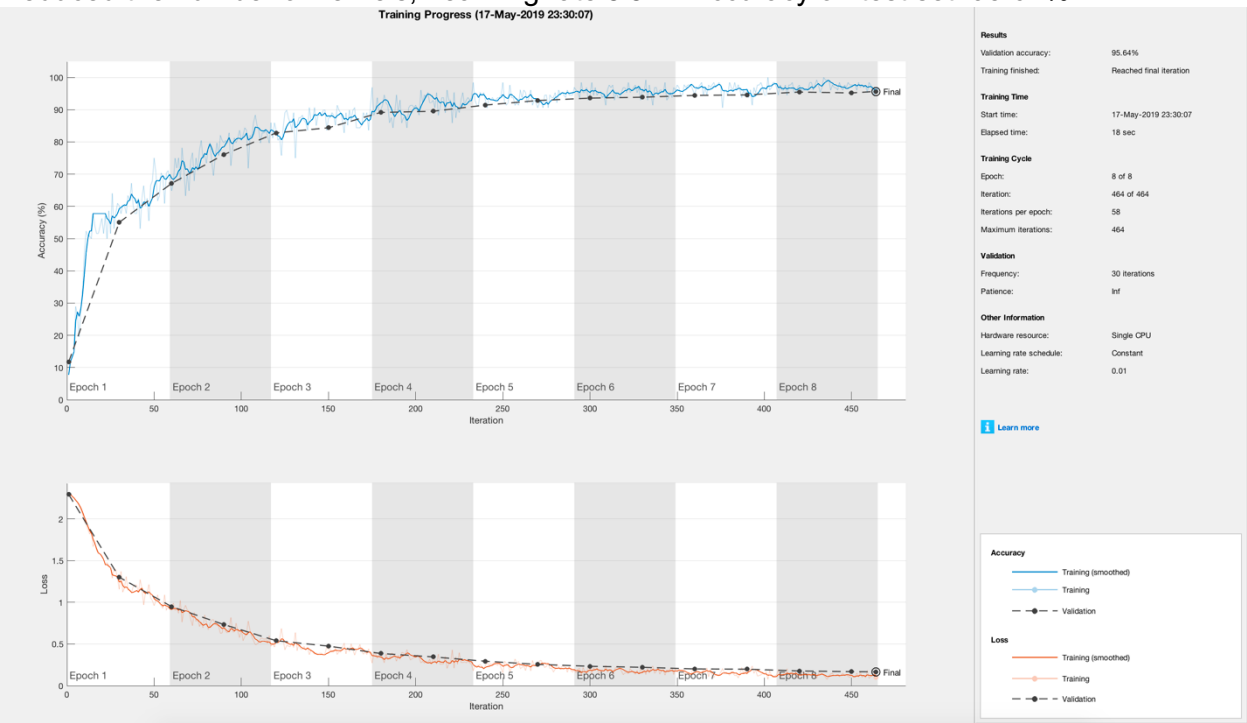
046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Section 8 – Less convolution kernels

Reduced the number of kernels, Learning rate **0.01** – Accuracy on test set: 95.64%



Reducing the number of convolution kernel implies a reduction in the number of learnable-parameters. As a result, the model becomes less “expressive” – meaning has limited learning capacity when compared with the architecture of section 6.

Overall, reducing the number of kernels led to lower accuracy and required additional epochs for training.

046746 Computer Vision – HW2

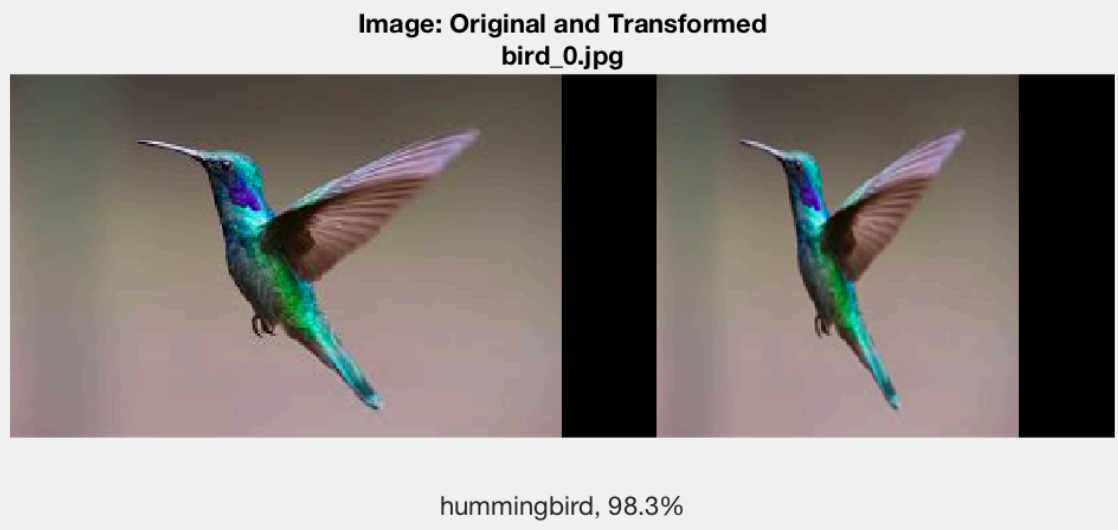
David Weinstein 302470596 dudiwa@campus.technion.ac.il

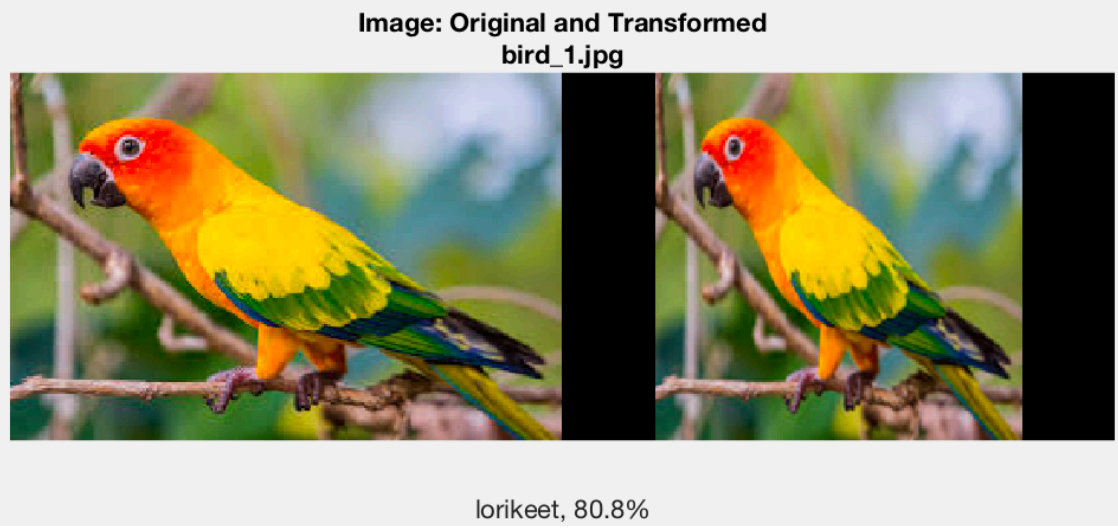
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Task B – Using Pre-trained network

Section 1-2 Load a pretrained VGG16 and classify images

The input images of birds (RGB) were resized to match the input layer of VGG16 $[182 \times 277 \times 3] \rightarrow [244 \times 244 \times 3]$ uint8. Note that the input layer applied a zerocenter normalization of the images. The transformed images were classified correctly:





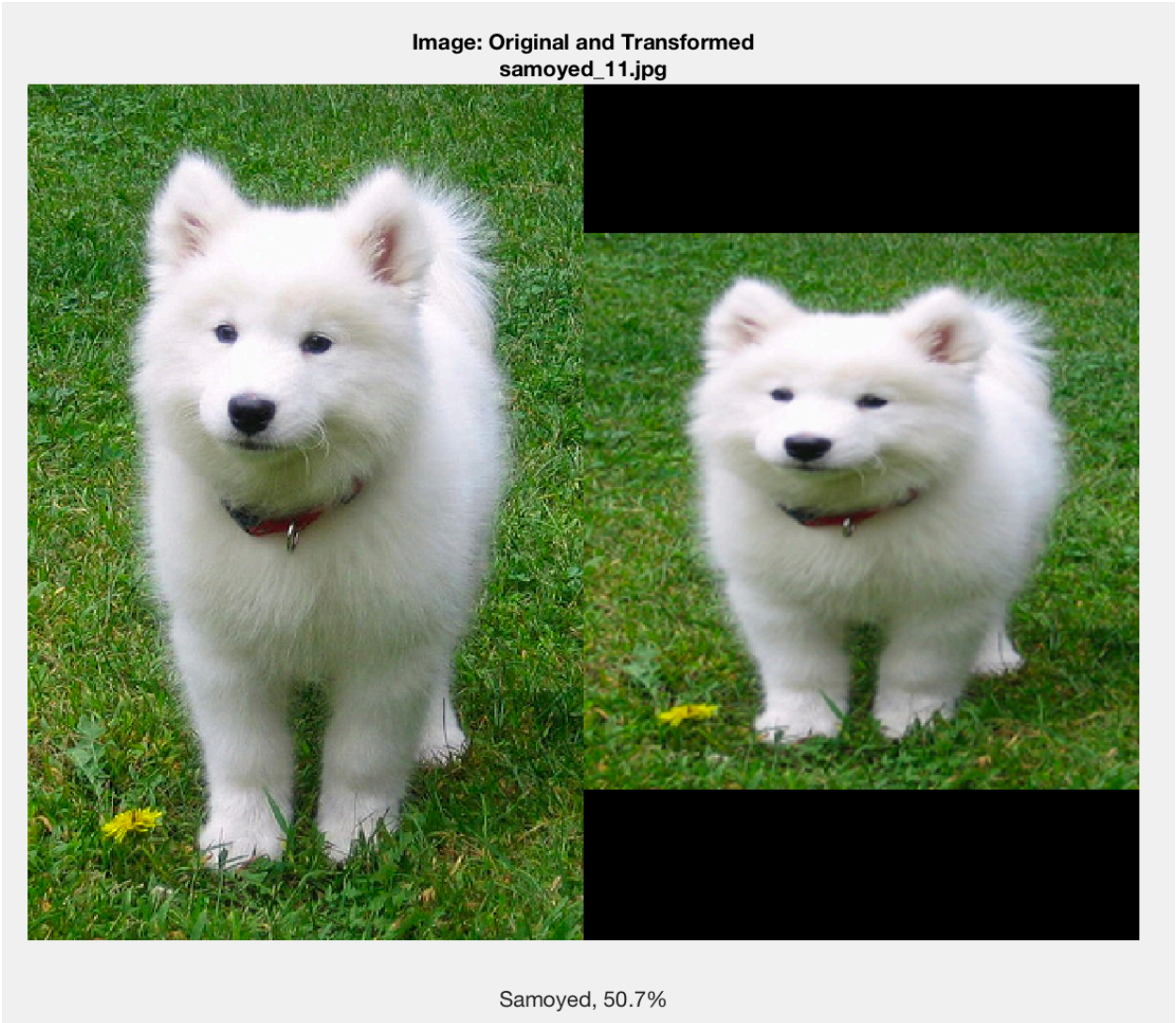
046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Section 3 – Classify an image (form the web)

Using an input image (RGB) of a Samoyed puppy, resized to fit the input layer of VGG16, the correct classification was obtained by the network.



046746 Computer Vision – HW2

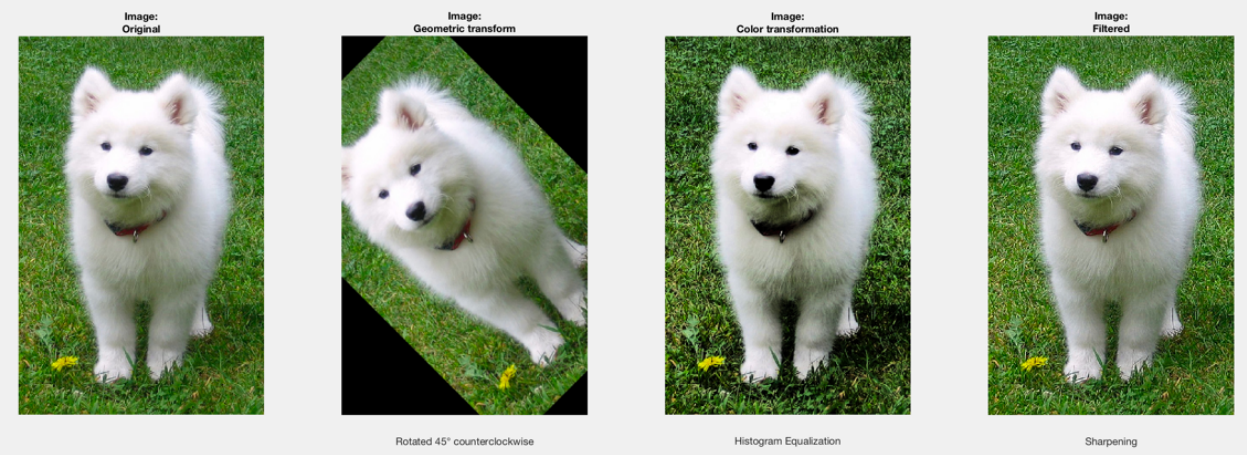
David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Section 4 – Transform the image and classify

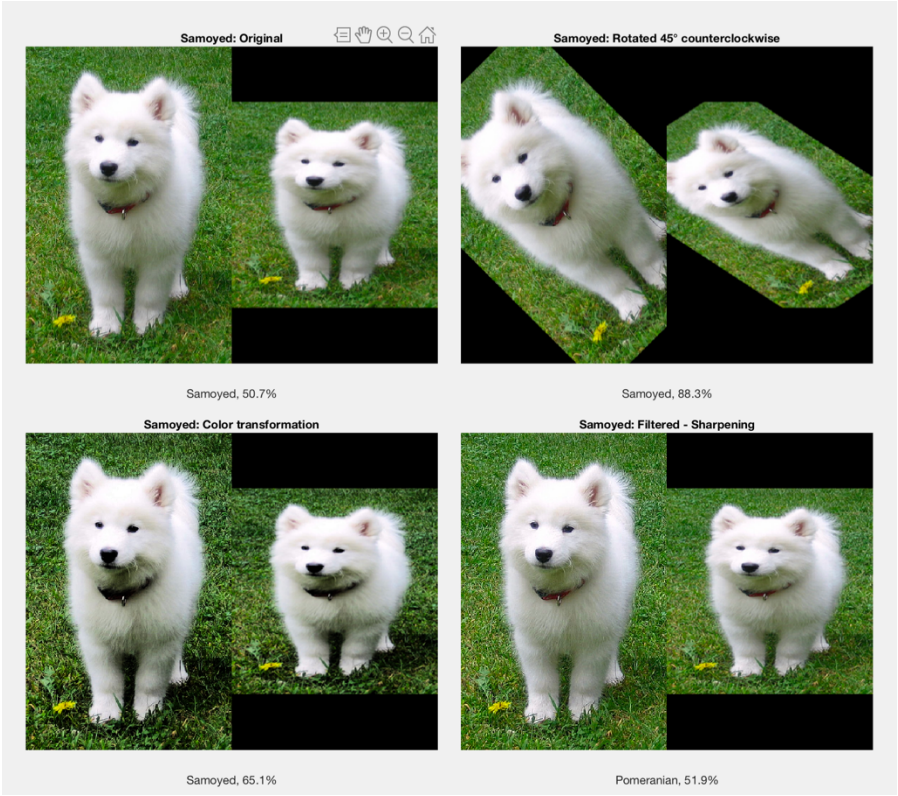
Using an input image (RGB) from section 3, the following alterations were applied:

- Geometric transformation – image **rotation** by 45 degrees counterclockwise
- Color transformation – **Histogram equalization** by converting to the HSV color space, equalizing the histogram of the V-channel (value) and converting back to RGB.
  The resulting image has increased global contrast, which is useful in an image with both bright and dark regions. This is noticeable by the pronounced shadow of the dog, and by the darker shade of its collar.
- Filter – **Sharpening** using "unsharp masking": subtract the gaussian blurred image from itself for enhanced-perceived sharpness (the grass leaves are more defined, and the fur is more detailed near the edges)



Classify with VGG16:

The histogram-equalized and rotated images were correctly classified (with a higher score than the original image).
The sharpened image correctly identified the category "dog" but misclassified the breed as Pomeranian ↓

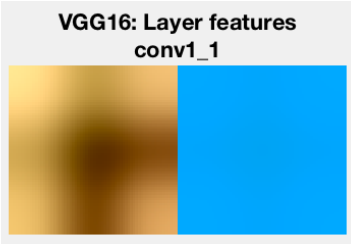


046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

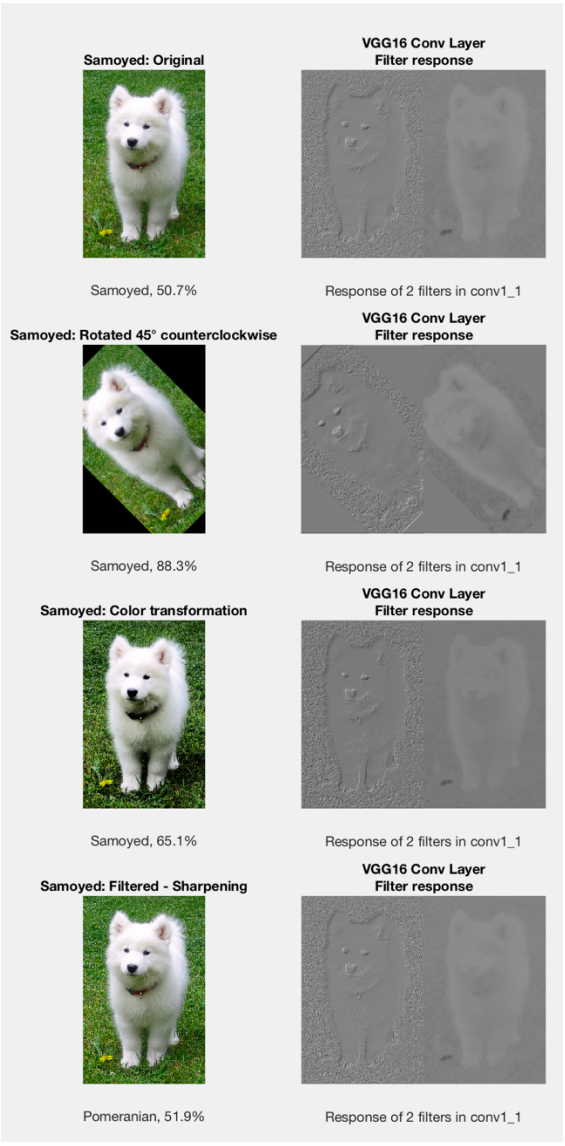
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Section 5 – Display 2 filters and their response

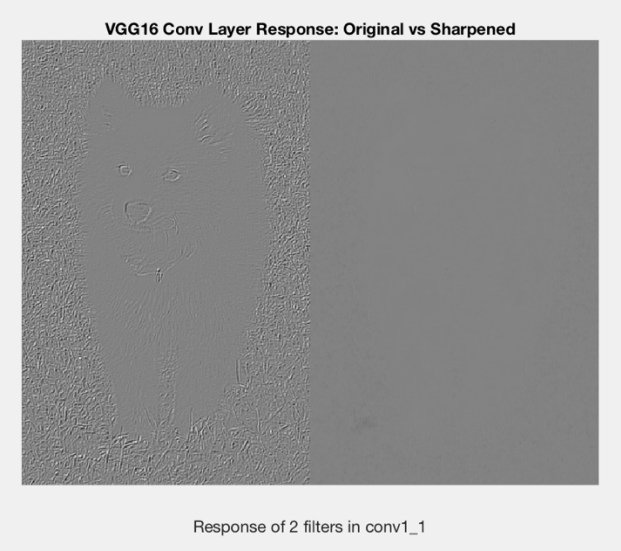


Two filters extracted from the first convolution layer **conv1_1**
The filters resemble a corner- template and a low-pass or “cartoon” response (produce boundaries and low-detail texture image).
The response to the images from section 4 is shown below:

The histogram-equalized and rotated images produce response with detailed features – similar to that of the original image.
The sharpened image contains pronounced edges, affecting the response and causing the network to respond strongly to the large number of edges which were introduced by the sharpening of the image.
As a result, the sharpened image was misclassified as a different breed (Saomyed → Pomeranian).



Below is the difference between the kernel response to the original image and the sharpened image (left – corner kernel, right – low pass kernel).



The large number of edges in the sharpened image impact the classification of the network, leading to misclassification of the dog’s breed.

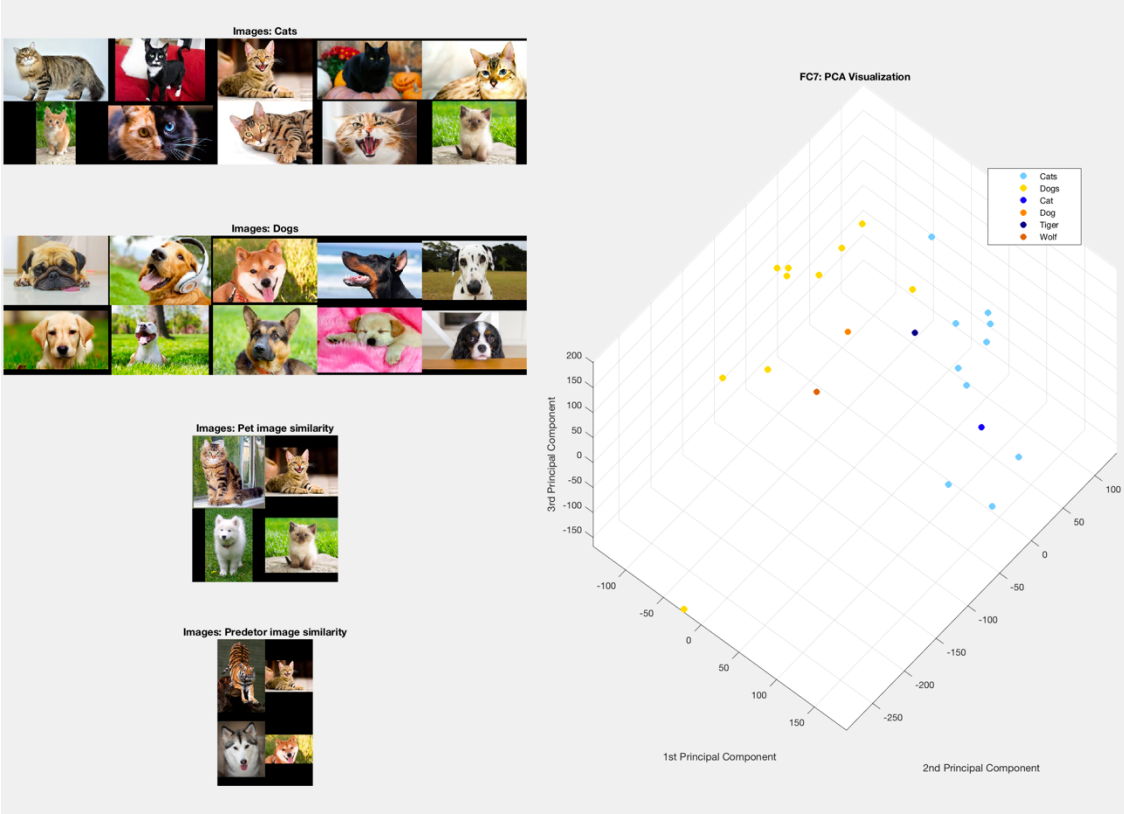
046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Section 6-8 – Fully connected (7) visualization

The figure below displays the “in”-data images of cats and dogs, and our own images of a cat, a dog, a tiger and a wolf. The network’s response to the images is visualized by extracting the feature vector of the fully-connected layer FC7 (of length 4096 per image) and projecting on to a submanifold of the feature space using PCA (corresponding to the 3 largest eigenvalues).



The “in”-data images produce two well-separated clusters of cats (light blue) and dogs (yellow).

The third row of images presents our own images of a cat (blue) and a dog (orange). Feature vectors of the network’s response are projected using PCA.
Using **L2** distance measure, the nearest “in”-data image is shown in the third row next to our custom image. As we can see, our cat shares similarity with its match from the “in”-data cat. However, our dog (a fluffy Samoyed puppy) was matched with one of the “in”-data cats. This match was consistent even with the **L1** distance measure (perhaps a larger dataset would help). The visual similarities make sense: both images show white long-hair and a grassy background.

Finally, we apply the process to an image of a tiger (deep blue) and a wolf (brown). The predator images are shown in the fourth row, next to their matching pet (“in”-data images). The predator feature-projection is in proximity to the correct “in”-data family, with the tiger matched with a cat with stripes and the wolf matched with a Shiba dog (similar nose).

046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Image Stitching

Using the SIFT algorithm to extract feature descriptors and frames of every given Stop Sign image (**section 1)**. By comparing the descriptors of the first image (that has been used as a template for Stop Sign) and the other images, we found key points of similarity between each pair of images (**section 2)**:

As we can see, some matching pairs are incorrect.
Because the affine transformation matrix (for projecting the first image to the second one) has six degrees of freedom – we require at least **3 corresponding pairs** of matching key-points. To cope with outliers, we can discard most of the key points. To do that, we build a function for finding the affine transform matrix (**section 3**) and for filtering the key points and finding the most suitable transform, we use the RANSAC algorithm function (**section 4)**. The remaining key-point pairs after RANSAC are shown below:

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Then, by applying the calculated affine transformation, we project the first image (**section 5**). This transforms the first image is shown below:

Finally, we stitched the projected template image to the second image (**section 6**):

The last results for the other images:

046746 Computer Vision – HW2

David Weinstein 302470596 dudiwa@campus.technion.ac.il

Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

As we can see, the last image yielded poor results after the stitching process. It is likely because the affine transformation matrix does not have enough degrees of freedom:

In the image above, the articulation of the stop-sign exhibits a tilt relative to the image plane. Thus, the affine transform is insufficient for adequately projecting the template unto the targets. Here we should opt for a projective transformation rather than an affine transformation.