1 Image preprocessing in our article

COVIDNet operates on authentic CXR images containing artifacts like additional metadata from medical equipment. Therefore the main motivation for preprocessing data is removing those artifacts and performing augmentation in order to account for a variety of technics behind creating CXR images. However preprocessing used by authors of COVIDNet is limited only to a handful of basic methods listed in the paper:

> (...) the chest CXR images were cropped (top 8% of the image) prior to training in order to mitigate commonly-found embedded textual information in the CXR images. Furthermore, to train the tested deep neural network architectures, data augmentation was leveraged with the following augmentation types: translation (± 10% in x and y directions), rotation (± 10), horizontal flip, zoom (± 15%), and intensity shift (± 10%)

The whole process is performed using file data.py, available at Github repository. Significant code chunks presented contain instructions for cropping and resizing images using arbitrary values for omission of the aforementioned artifacts and performing basic random transformations like zooming and rotating. This phase of preprocessing and augmentation is performed 'just-in-time' during the model training and testing.

```
def crop_top(img, percent=0.15):
    offset = int(img.shape[0] * percent)
    return img[offset:]

def central_crop(img):
    size = min(img.shape[0], img.shape[1])
    offset_h = int((img.shape[0] - size) / 2)
    offset_w = int((img.shape[1] - size) / 2)
    return img[offset_h:offset_h + size, offset_w:offset_w + size]

def process_image_file(filepath, top_percent, size):
    img = cv2.imread(filepath)
    print(filepath)
    print(type(img))
    img = crop_top(img, percent=top_percent)
    img = central_crop(img)
    img = cv2.resize(img, (size, size))
    return img
```

Figure 1: Chunk from data.py - crop functions

```python
def random_ratio_resize(img, prob=0.3, delta=0.1):
    if np.random.rand() >= prob:
        return img
    ratio = img.shape[0] / img.shape[1]
    ratio = np.random.uniform(max(ratio - delta, 0.01), ratio + delta)

    if ratio * img.shape[1] <= img.shape[1]:
        size = (int(img.shape[1] * ratio), img.shape[1])
    else:
        size = (img.shape[0], int(img.shape[0] / ratio))

    dh = img.shape[0] - size[1]
    top, bot = dh // 2, dh - dh // 2
    dw = img.shape[1] - size[0]
    left, right = dw // 2, dw - dw // 2

    if size[0] > 480 or size[1] > 480:
        print(img.shape, size, ratio)

    img = cv2.resize(img, size)
    img = cv2.copyMakeBorder(img, top, bot, left, right, cv2.BORDER_CONSTANT,
                             (0, 0, 0))

    if img.shape[0] != 480 or img.shape[1] != 480:
        raise ValueError(img.shape, size)
    return img
```

Figure 2: Chunk from data.py - random ratio changes

Additional initial preprocessing is performed earlier during combining multiple repositories into the one basic dataset. There images from specific datasets are being converted into the gray scale.

No further advanced preprocessing is performed.

```
if patient[0] in test_patients:
    if patient[3] == 'sirm':
        image = cv2.imread(os.path.join(ds_imgpath[patient[3]], patient[1]))
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        patient[1] = patient[1].replace(' ', '')
        cv2.imwrite(os.path.join(savepath, 'test', patient[1]), gray)
```

Figure 3: Example of code chunk converting imported image to the gray scale

2 Image preprocessing modifications in our project

We are facing problems in working with architecture of our neural network. Due to lack of code of the architecture and lack of skills in working with tensorflow and networks we were not able to provide second homework yet and it has influenced the quality of this task which is part of 3rd homework. We are hoping to get some extra time to finish homework 2 and present more advanced results of modifications of data augmentatnion. If it is possible we will try to solve this issues during the spring break.
However, we will partially present our efforts and we will still work for improvement.

We used two main approaches:

1. Fourier transformation
2. subtracting mean pixel value

Both approaches during our initial assesment proved to be highly innefective in comparison with plane images used by COVIDNet authors. Especially Fourier transform, considering authors didn't provide detailed architecture, but pretrained models.

2.1 Fourier

The following chunk of code has been added to data preprocessing part.

We have tried to convert the image into frequency space and then apply the filter. Many tries to feed the network with image in this form caused errors. Only after bringing image back to the original space we did manage to start training - but the results are not what we expected.

```
x = cv2.cvtColor(x, cv2.COLOR_BGR2GRAY)

rows, cols = x.shape[0], x.shape[1]
crow, ccol = rows // 2, cols // 2

dft = cv2.dft(np.float32(x), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
#magnitude_spectrum = 20 * np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows, cols, 2), np.uint8)
mask[crow - 230:crow + 230, ccol - 230:ccol + 230] = 1
# apply mask and inverse DFT
fshift = dft_shift * mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])

x = img_back

x = cv2.cvtColor(x, cv2.COLOR_GRAY2BGR)
```

Figure 4: Fourier transform code

Not applying the mask at all did not help
We supsect some lack of understanding in doing Fourier Transform what caused this poor results, however, we weren't able to spot the mistake yet - more intuitive approches to feed iamges have caused errors with number of channels and we couldn't fix it yet.

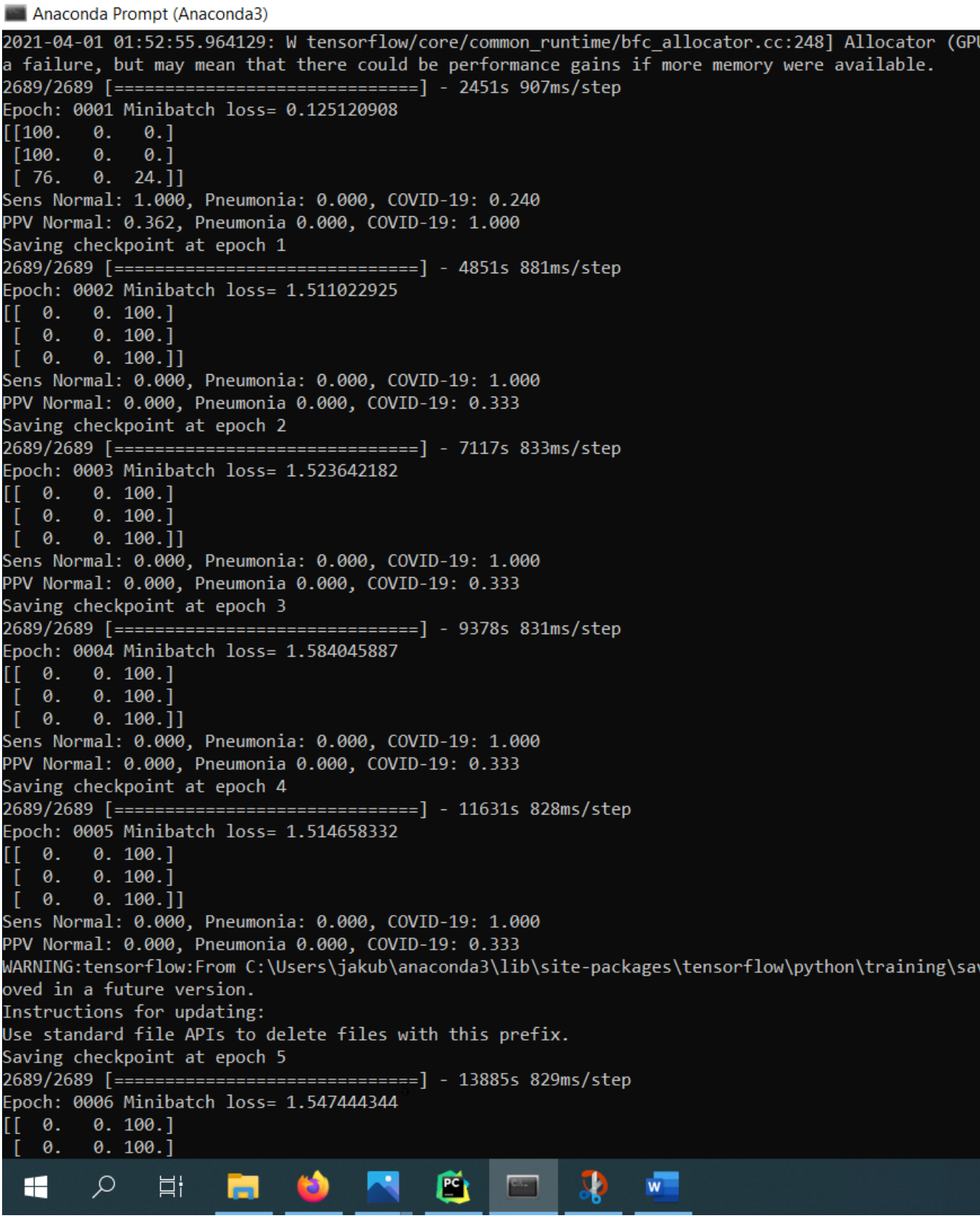


Figure 5: Fourier transform code

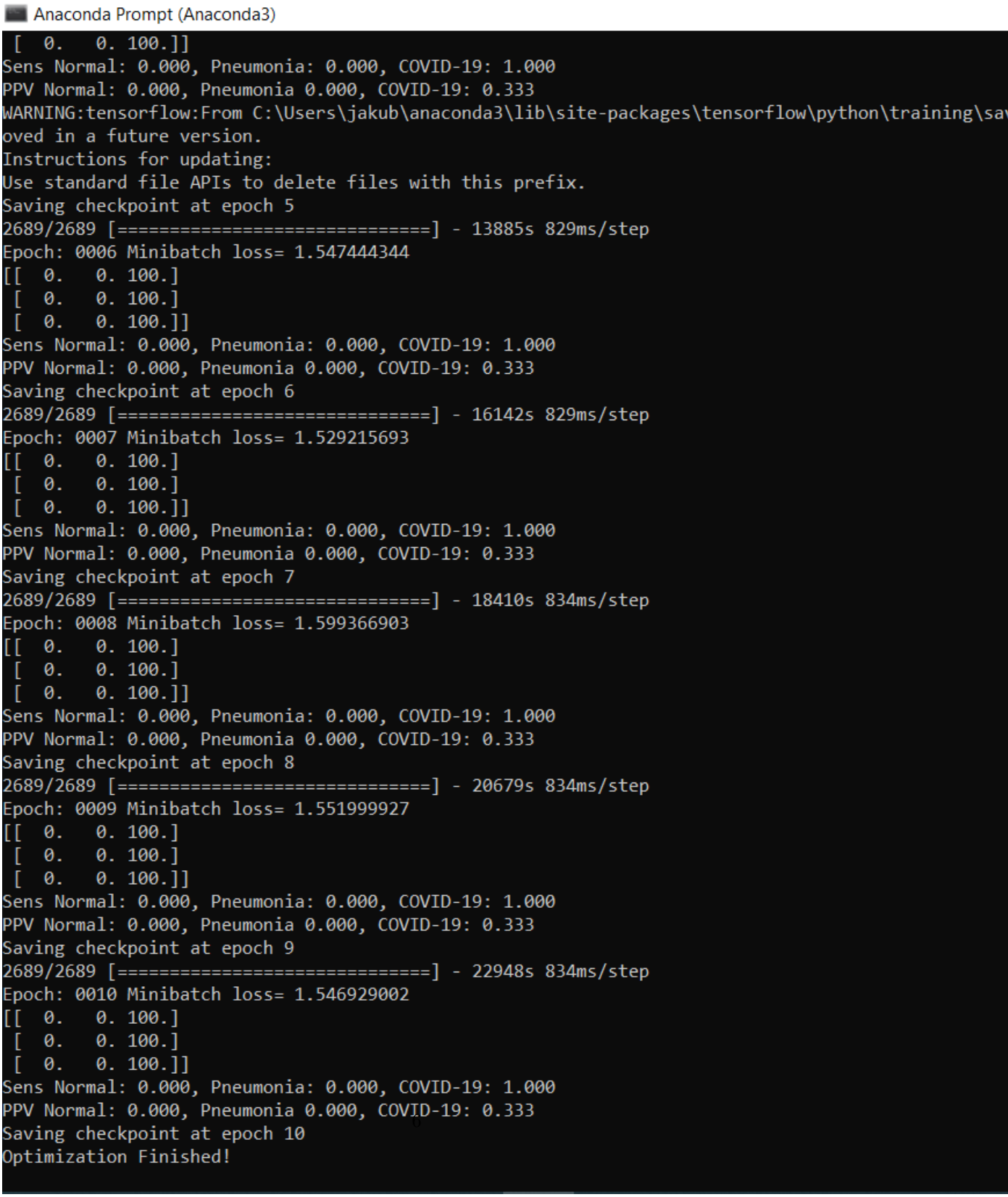


Figure 6: Fourier transform code

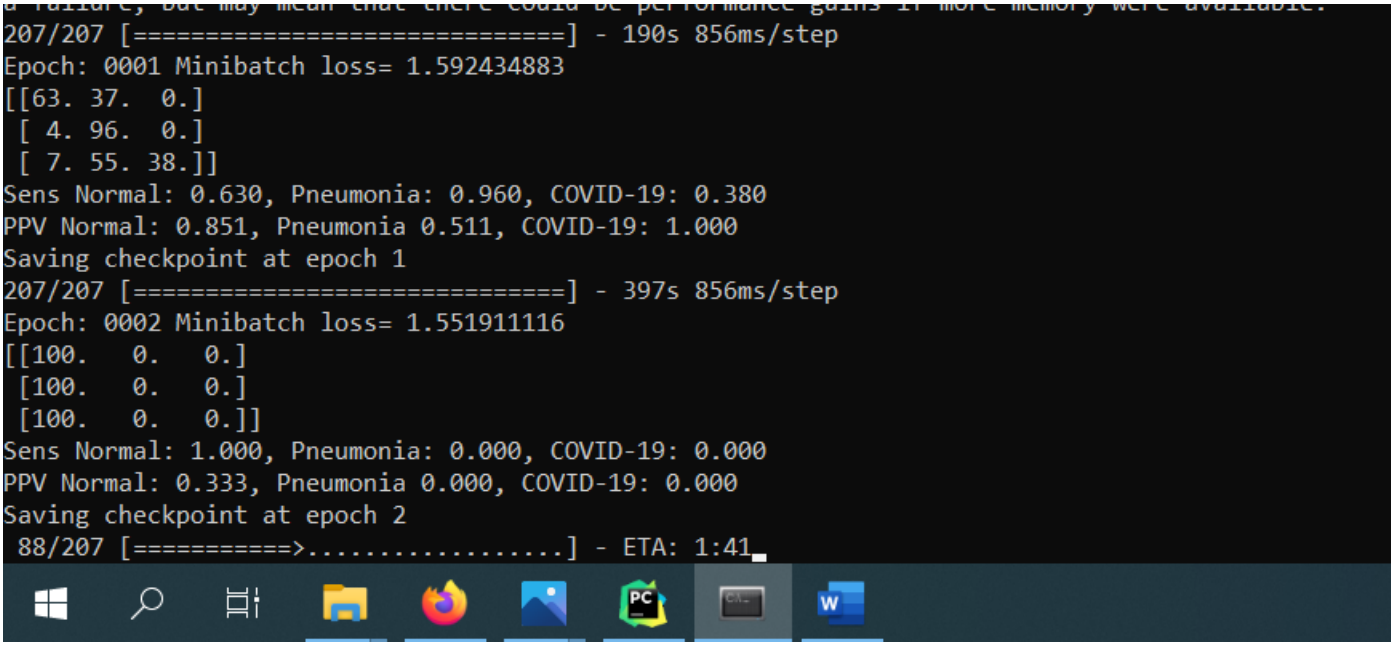


Figure 7: Fourier transform code

3 Methods of image preproscessing for objects detection

Methods may differ depending on input image type and feature extraction, however, let's take a brief look on the most common ones.

- smoothing images
  Smoothing (also called blurring) is done for different reason. It becomes really useful when it comes to image denoising. Images may contain some incorrect information, even though the cameras are getting better and better. Aim of denosinig is to properly identificate the noise and remove it in a way that does not damage the image structure of contained information. OpenCV package offers various filters, that we may applay depeding on our needs - normalized box filter, Gaussian filter, median filter, biltaeral filter

- contrast enhancement
  Grayscaled image may be too bright or too dark, thus a method called histogram equalization may be applied. This method is about stretching out the intensity range. Look at the example from openCV, where image with pixels clustered around certain intensity is being processed.

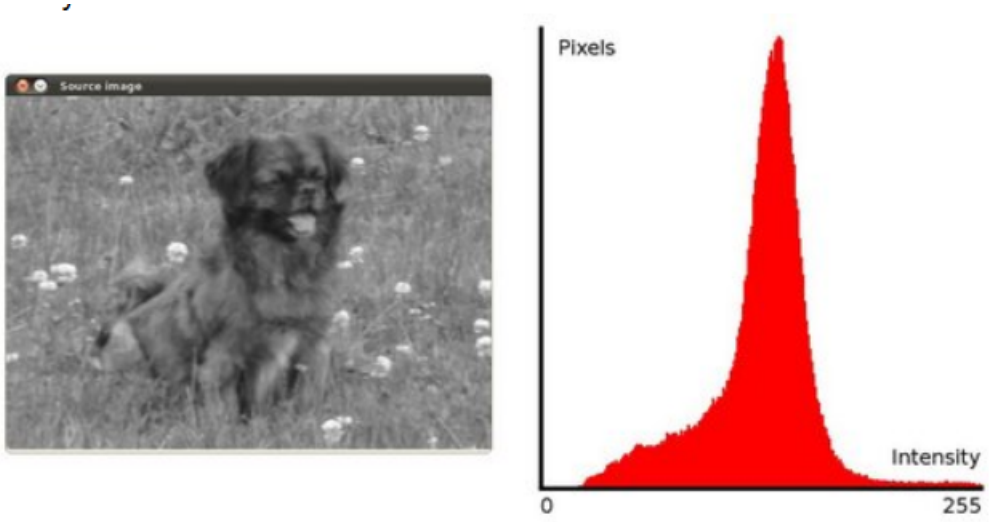


Figure 8: Histogram equalization - 1

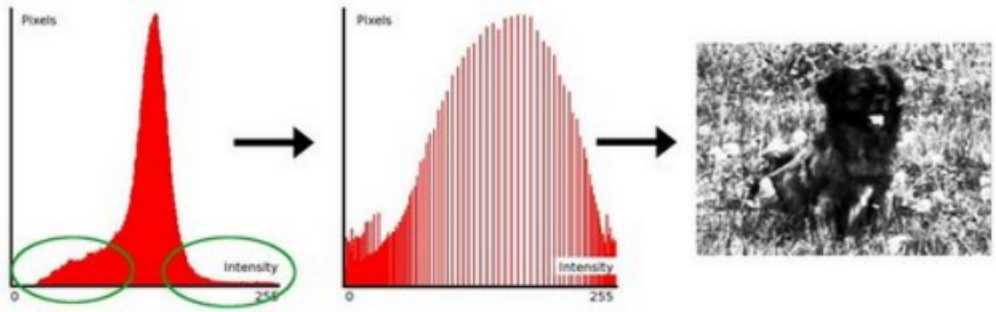
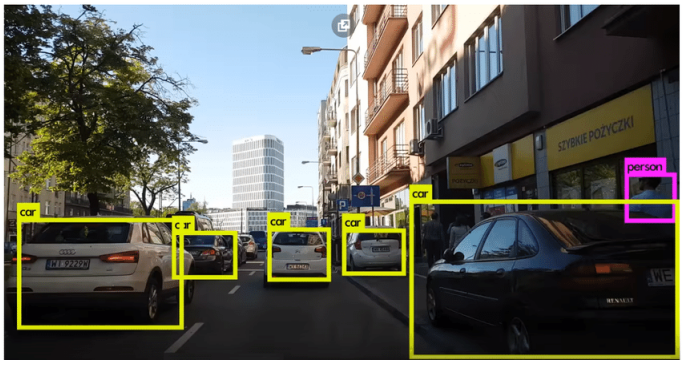


Figure 9: Histogram equalization - 2

- restoration
  This method is used for reconstruction of destroyed or missing parts of an image.
- Circural Hough Transforom
  Aim of this method is to find circular patterns within an image. "CHT is used to transform a set of feature points in the image space into a set of accumulated votes in a parameter space. Then, for each feature point, votes are accumulated in an accumulator array for all parameter combinations. The array elements that contain the highest number of votes indicate the presence of the shape. *Rizon, Mohamed, et al. "Object detection using circular Hough transform." (2005).*
- morphologcial operations
  It is set of operations used to process images based on shapes. It is commonly used for

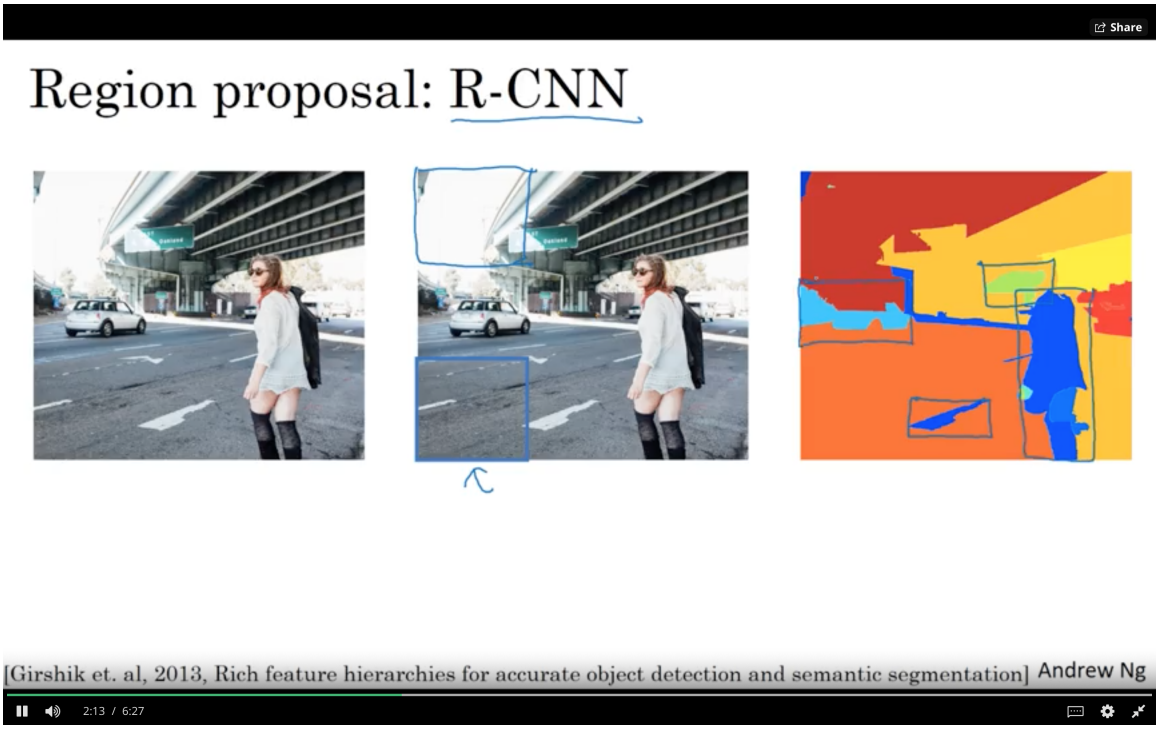
removing noise, isolation of individual elemetns and joining disparate elements in an image, finding intensity bumps or holes in image. The most basic operations are Erosion and Dilation.

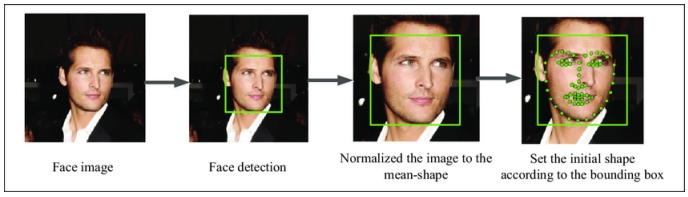
- YOLO algorithm
  This is popular object detection algorithm. However, the data images must be prepocess in order to work properly. Firstly, image examples need to be resize to the same shape. This process is natural to be done since algortihm uses so call anchor boxes of a fixed shape. Then we uses that anchor boxes to look for the objects. This leeds us to another importatnt step which is trying to keep the object from the same class in the relatively same size. Also chosing the right size of anchor box is the parameter than we have to learn.



- Segmentation algorithm(R-CNN)
  This is a better version of detection algortihm where you just move your box hoping to find a class obejct. The method proposed by Russ Girshik, Jeff Donahue, Trevor Darrell, and Jitendra Malik chose one those boxes that have potentail to have class object in it. This is done by segmantation photo on regions.

- Face veryfication and recognition (One shot learning algorithm) Oftentimes whe have to face the situation where we do not have big amount of data. However we would like to to verify someone for exampl by his Id photo. Nowaday we can find such a solution on airport. The idea by perprocesing is that the algorithm has to find our face in the photo and then cut it form the picture so that the neutral network only get relavant information to its input.





4 Methods used in increasing the resolution

INTERPOLATION METHODS

- Nearest neighbour
  Firstly new pixels are created. Number of new pixels depends on the size of enlargement. Then colours have to be assigned. In this method colour of the nearest pixel in original photo is examined and then assigned to new location. This method is simple and fast, however, have many downfalls. It produes a checkerboard effect.

Figure 10: Checkerboard effect

- Bilinear Interpolation
  Destination colour is computed from the top, left, right and bottom pixel of its original position. Then readings of colours are added and divided by 4

- Bicubic Interpolation
  This method uses information from an original pixel and sixteen of the surrounding pixels. This method is more sophisticated than 2 mentioned before. It uses information from larger number of pixels and computations are made in other way - with usage of Bicubic calculation.

  Probably this method is most commonly used.

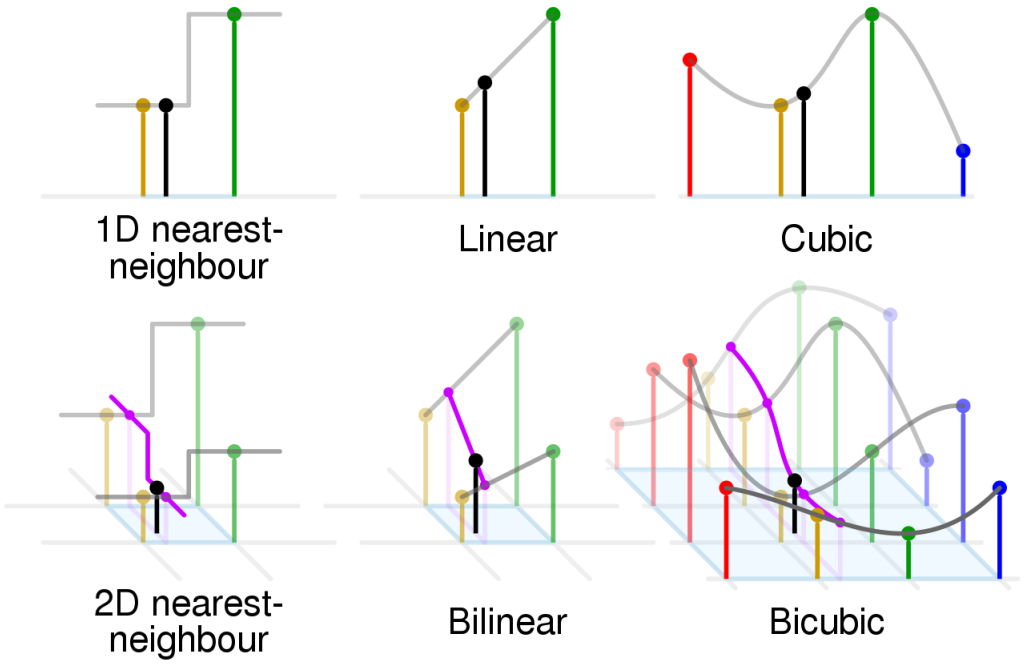


Figure 11: Comparison of Bicubic interpolation with some 1- and 2-dimensional interpolations.

Summary:
Nearest Neighbor - gives jaggies or stair-case effect.
Bilinear - quite fast but worse than Bicubic.
Bicubic - slowest but produces best estimation of new pixel valeus.
Those methods which are based on interpolation result in blurred image. Artifacts are especially present in areas with sharp lines and distinct color changes.

PURPOSE OF INCREASING RESOLUTION
1. it can provide better detailed zooming and magnification of details in an image.
2. conversion of low-resolution digital images and videos to a higher resolution.
3. Resolution increasing and decreasing can be used in sending images over a network, instead of compression.

Color shifting. Both this is an augmenation process and increasing resolution. Sometimes the colors in the picture are very weak or dark. This procedure helps for example to brightend the photo or balanced the color resolution