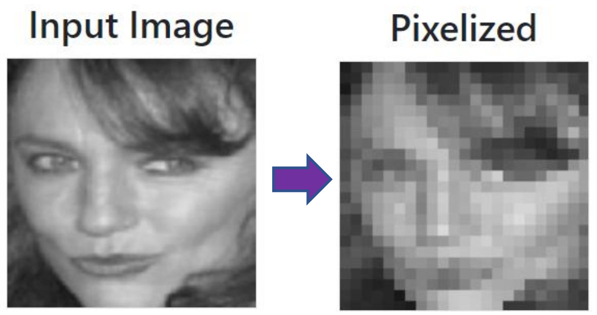
Homework: Face De-Identification & Its Attacks and Defenses

Version: 1

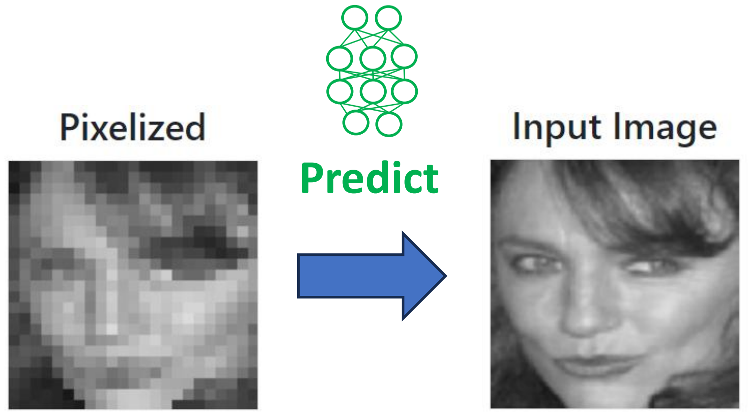
Multimedia Security

- **Step 1: Make face de-identification by the two methods, “blurring” and “pixelization”**
  - Some resources
    - https://pyimagesearch3.rssing.com/chan-55716380/all_p15.html
    - https://www.geeksforgeeks.org/how-to-blur-faces-in-images-using-opencv-in-python/
    - https://github.com/understand-ai/anonymizer
    - https://medium.com/lifes-a-struggle/hog-svm-c2fb01304c0



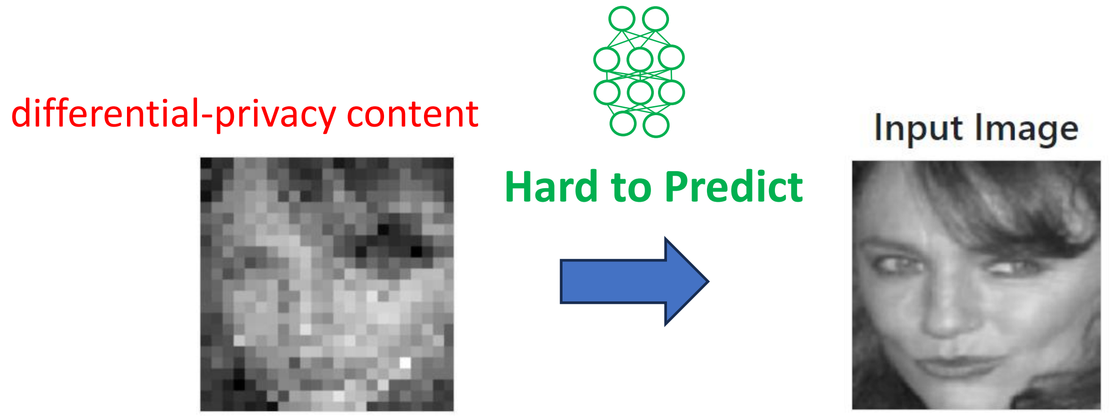
Multimedia Security

- **Step 2: Attack this face de-identification system by AI methods.**
  Try to recognize/predict the original face, even after it has been de-identified.
  - Some resources
    - https://arxiv.org/pdf/1609.00408.pdf
    - https://petsymposium.org/popets/2016/popets-2016-0047.pdf



Multimedia Security

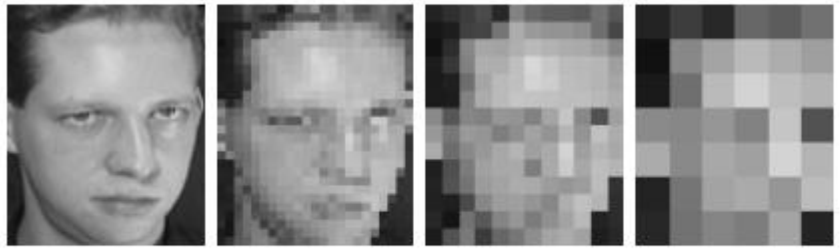
- **Step 3: Defend against the above AI attack model by "differential privacy"**
  - You may attempt this, *with or without re-training* the model, using differential privacy techniques.
  - Some resources
    - https://webpages.charlotte.edu/lfan4/pdf/TPDP2019.pdf
    - http://3.223.148.187/methods
    - https://fan-group.github.io/imageprivacy/
    - https://hal.inria.fr/hal-01954420/file/470961_1_En_10_Chapter.pdf



Some Examples for Possible Result Format

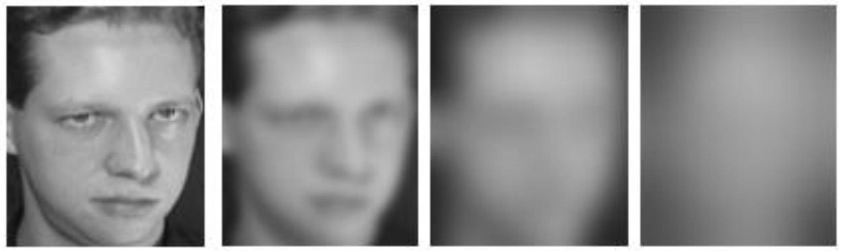
(Example) Possible Result Format of Step 1

- Pixelization
  - Different *pixel* size   v.s.   visual quality
  - e.g., For each cell containing *b* x *b* pixels, a smaller *b* value yields better visual quality.



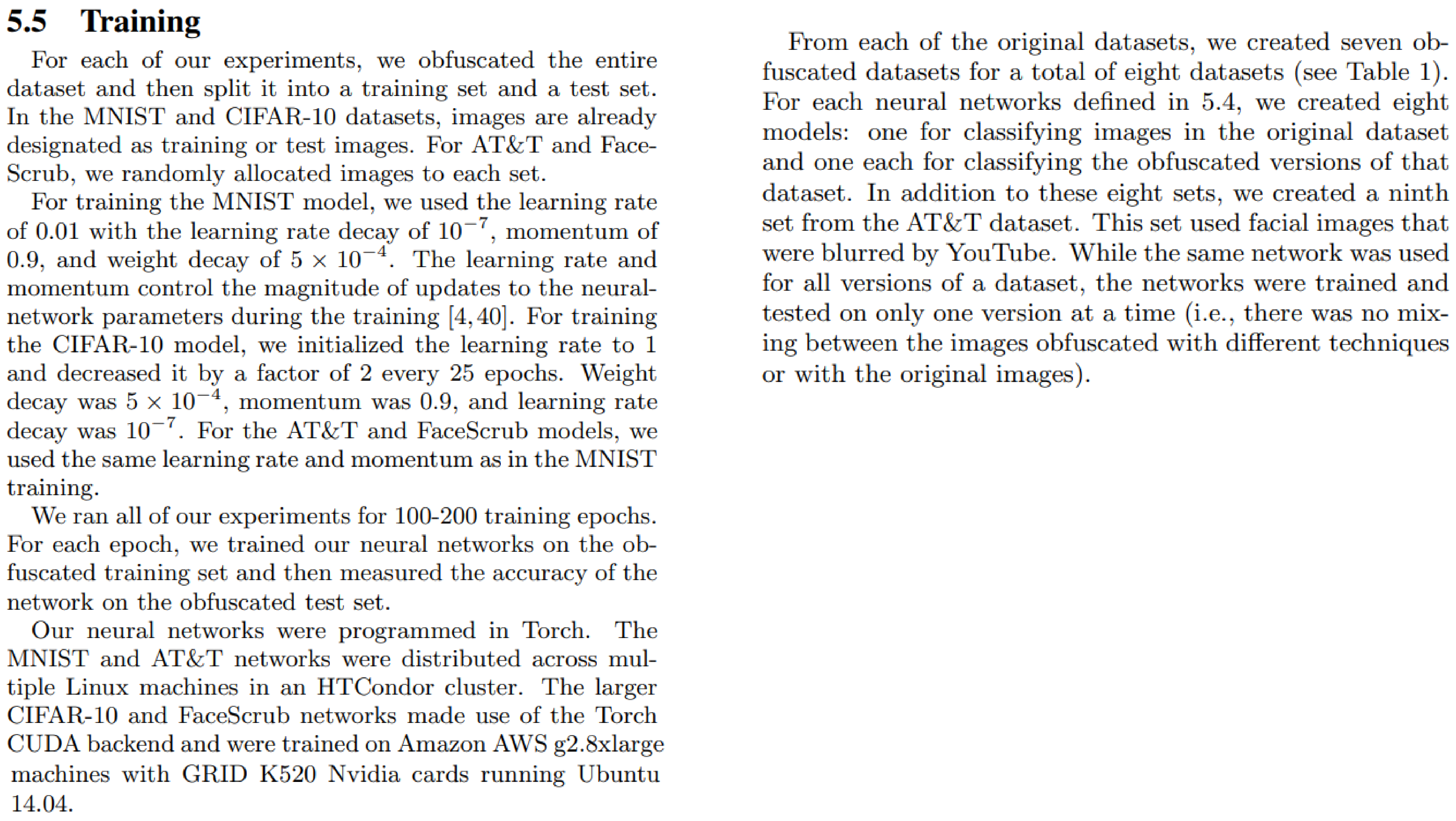
(a) orig   (b) b=4   (c) b=8   (d) b=16

- Gaussian blur
  - Different Gaussian parameters   v.s.   visual quality
  - e.g., For each pixel, a *k* x *k* neighborhood around the pixel is considered and the *gaussian weighted average* is released. In general, a larger *k* value yields lower visual quality by over smoothing the image
  - More parameters can be found here: [OpenCV: Smoothing Images]



(a) orig   (e) k=15   (f) k=45   (g) k=99

(Example) Possible Result Format of Step 2 (1)



5.5 Training

For each of our experiments, we obfuscated the entire dataset and then split it into a training set and a test set. In the MNIST and CIFAR-10 datasets, images are already designated as training or test images. For AT&T and FaceScrub, we randomly allocated images to each set.

For training the MNIST model, we used the learning rate of 0.01 with the learning rate decay of 10^{-7}, momentum of 0.9, and weight decay of 5×10^{-4}. The learning rate and momentum control the magnitude of updates to the neural-network parameters during the training [4,40]. For training the CIFAR-10 model, we initialized the learning rate to 1 and decreased it by a factor of 2 every 25 epochs. Weight decay was 5×10^{-4}, momentum was 0.9, and learning rate decay was 10^{-7}. For the AT&T and FaceScrub models, we used the same learning rate and momentum as in the MNIST training.

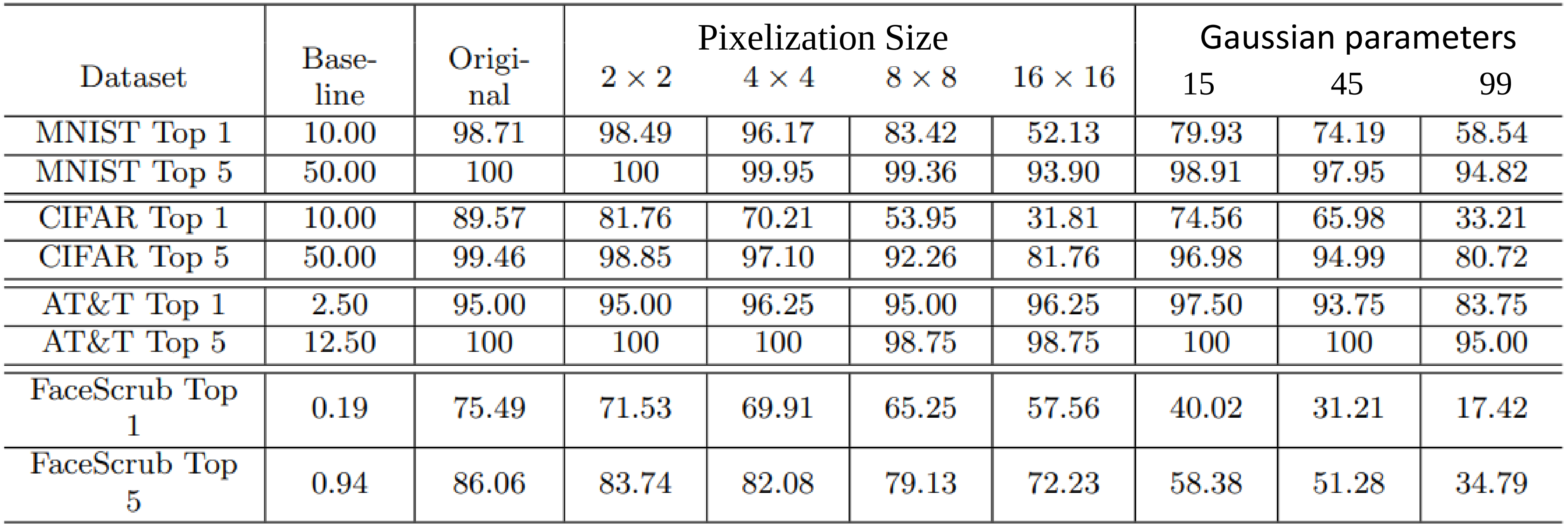
We ran all of our experiments for 100-200 training epochs. For each epoch, we trained our neural networks on the obfuscated training set and then measured the accuracy of the network on the obfuscated test set.

Our neural networks were programmed in Torch. The MNIST and AT&T networks were distributed across multiple Linux machines in an HTCondor cluster. The larger CIFAR-10 and FaceScrub networks made use of the Torch CUDA backend and were trained on Amazon AWS g2.8xlarge machines with GRID K520 Nvidia cards running Ubuntu 14.04.

From each of the original datasets, we created seven obfuscated datasets for a total of eight datasets (see Table 1). For each neural networks defined in 5.4, we created eight models: one for classifying images in the original dataset and one each for classifying the obfuscated versions of that dataset. In addition to these eight sets, we created a ninth set from the AT&T dataset. This set used facial images that were blurred by YouTube. While the same network was used for all versions of a dataset, the networks were trained and tested on only one version at a time (i.e., there was no mixing between the images obfuscated with different techniques or with the original images).

(Example) Possible Result Format of Step 2 (2)

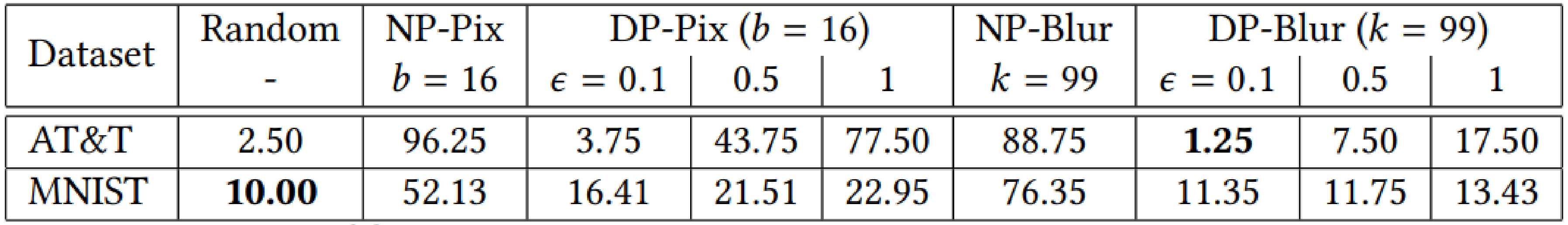
- Accuracy of neural networks classifying the original datasets as well as those obfuscated with different Pixelization size and Gaussian parameters.
  - The baseline accuracy corresponds to random guessing.



Dataset	Base-line	Origi-nal	Pixelization Size				Gaussian parameters		
			2×2	4×4	8×8	16×16	15	45	99
MNIST Top 1	10.00	98.71	98.49	96.17	83.42	52.13	79.93	74.19	58.54
MNIST Top 5	50.00	100	100	99.95	99.36	93.90	98.91	97.95	94.82
CIFAR Top 1	10.00	89.57	81.76	70.21	53.95	31.81	74.56	65.98	33.21
CIFAR Top 5	50.00	99.46	98.85	97.10	92.26	81.76	96.98	94.99	80.72
AT&T Top 1	2.50	95.00	95.00	96.25	95.00	96.25	97.50	93.75	83.75
AT&T Top 5	12.50	100	100	100	98.75	98.75	100	100	95.00
FaceScrub Top 1	0.19	75.49	71.53	69.91	65.25	57.56	40.02	31.21	17.42
FaceScrub Top 5	0.94	86.06	83.74	82.08	79.13	72.23	58.38	51.28	34.79

(Example) Possible Result Format of Step 3 (1)

- Accuracy of neural networks classifying the original datasets as well as those traditional obfuscated with different Pixelization size and Gaussian parameters, and *differentially-private* obfuscation with different ϵ.



Dataset	Random	NP-Pix	DP-Pix ($b = 16$)			NP-Blur	DP-Blur ($k = 99$)		
	-	$b = 16$	$\epsilon = 0.1$	0.5	1	$k = 99$	$\epsilon = 0.1$	0.5	1
AT&T	2.50	96.25	3.75	43.75	77.50	88.75	**1.25**	7.50	17.50
MNIST	**10.00**	52.13	16.41	21.51	22.95	76.35	11.35	11.75	13.43

Table 1: Accuracy (in %) of CNN Re-Identification Attacks

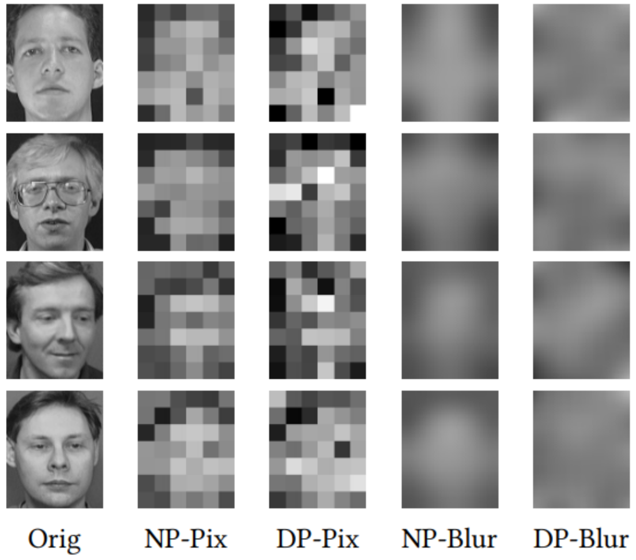
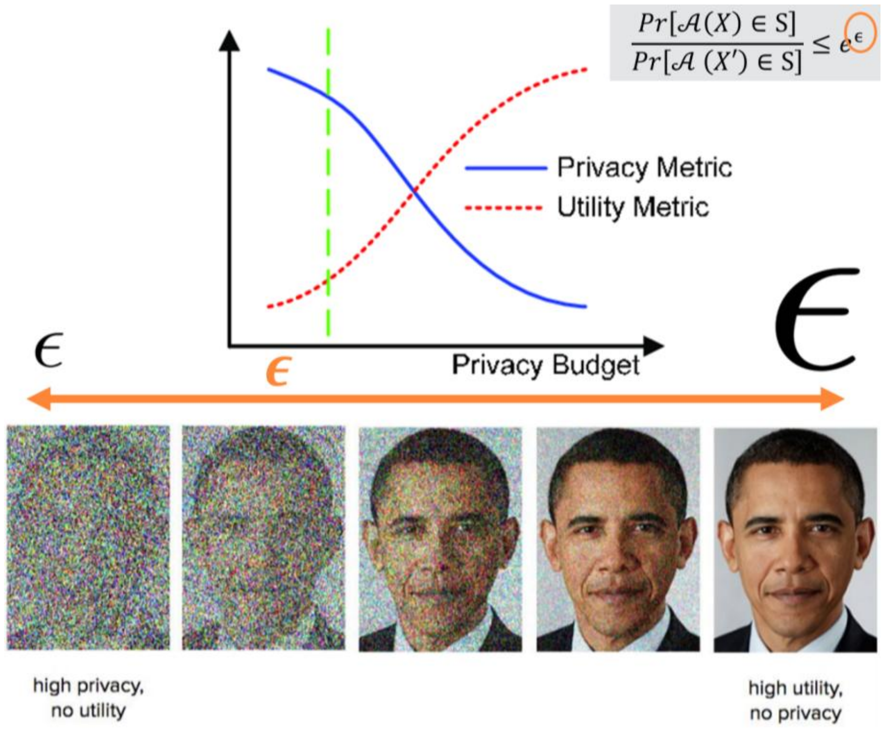


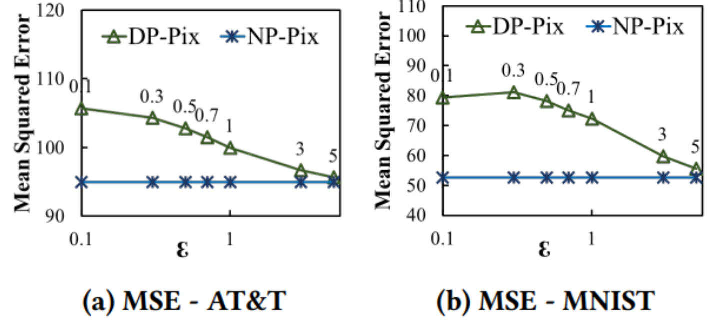
Table 2: Qualitative utility of differentially private obfuscation: each column represents one obfuscation method.

(Example) Possible Result Format of Step 3 (2)

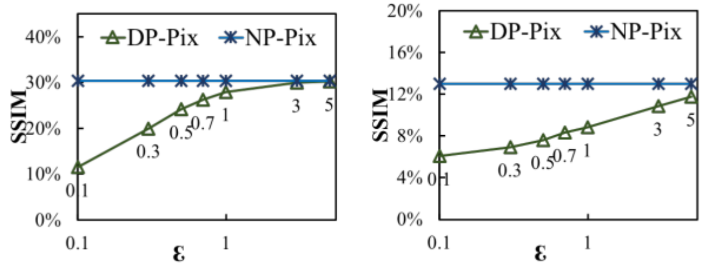
- The utility of the obfuscated image can be measured by:
  - Standard Mean Square Error (MSE) v.s. differential-privacy parameter ϵ
  - Structural Similarity (SSIM) v.s. differential-privacy parameter ϵ
    - How to calculate the Structural Similarity Index (SSIM) between two images with Python

- Notice: For **smaller** ϵ, it should have stronger obfuscated degree.



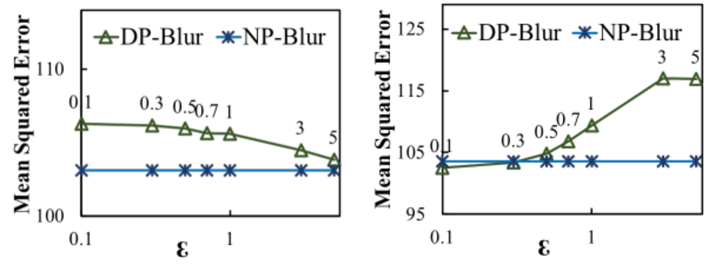


(a) MSE - AT&T (b) MSE - MNIST

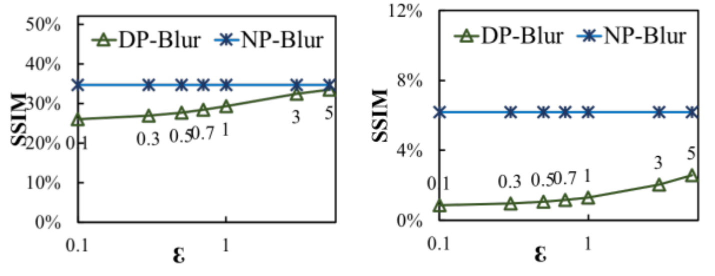


(c) SSIM - AT&T (d) SSIM - MNIST

Figure 3: Utility of *pixelization* versus privacy ϵ



(a) MSE - AT&T (b) MSE - MNIST



(c) SSIM - AT&T (d) SSIM - MNIST

Figure 4: Utility of *blurring* versus privacy ϵ

Upload the result to Moodle

- Upload your code and the ***detailed*** reports to Moodle.
  - **Code**: You must add some comments in your code for easy understanding.
  - **Report Files**: Document (Word).
    - Including a table in your report outlining the responsibilities of each team member for this homework.
  - Zipped to a file. The file name should be “TeamName_HW3.zip”