פרויקט DL

רחלי אליהו 316462324 | ספיר נחום 205651318

Epoch size: 150

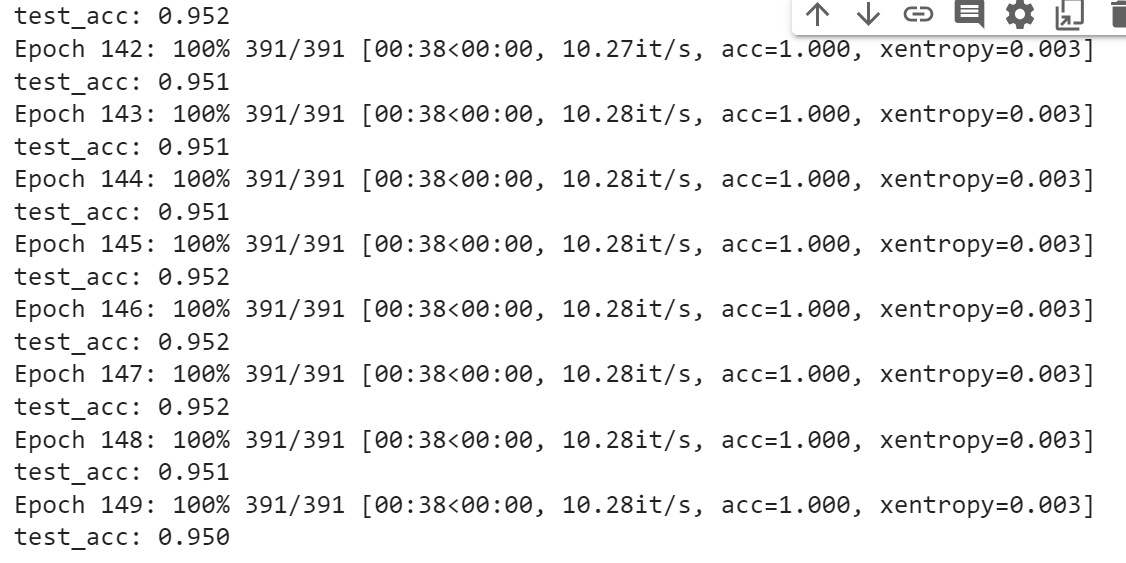
Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.RandomHorizontalFlip())

model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --shape triangle --length 16 --epochs 150



Epoch size: 150

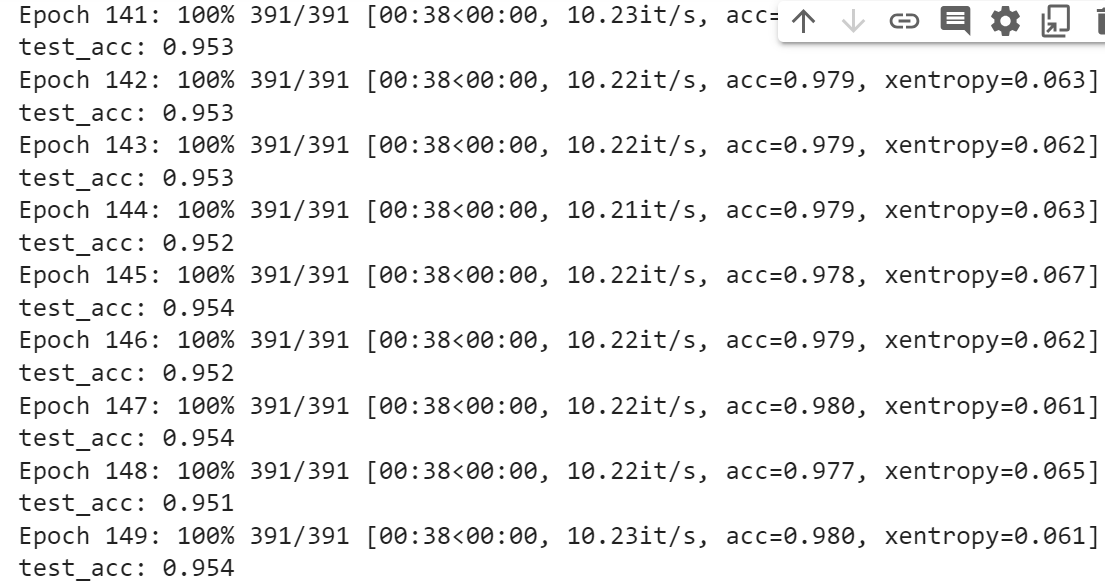
Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.RandomHorizontalFlip())

model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape square --length 16 --epochs 150



Epoch size: 150

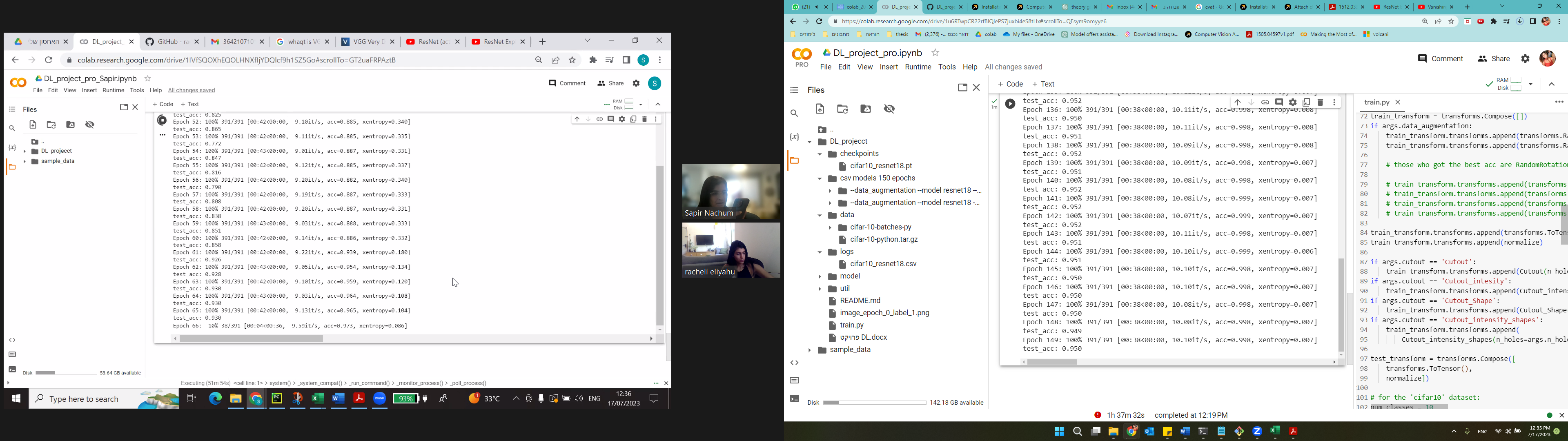
Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.RandomHorizontalFlip())

Model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape circle --length 16 --epochs 150



Epoch size: 150

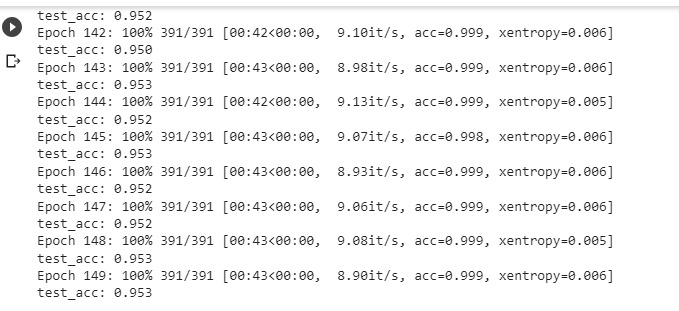
Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.RandomHorizontalFlip())

Model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape triangle --length 16 --epochs 150



Epoch size: 150

Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.RandomHorizontalFlip())

Model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_intensity\_shapes --shape square --length 16 --epochs 150

A screenshot of a computer

Description automatically generated

Epoch size: 150

Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.RandomHorizontalFlip())

Model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_intensity\_shapes --shape circle --length 16 --epochs 150

A screenshot of a computer

Description automatically generated

Epoch size: 150

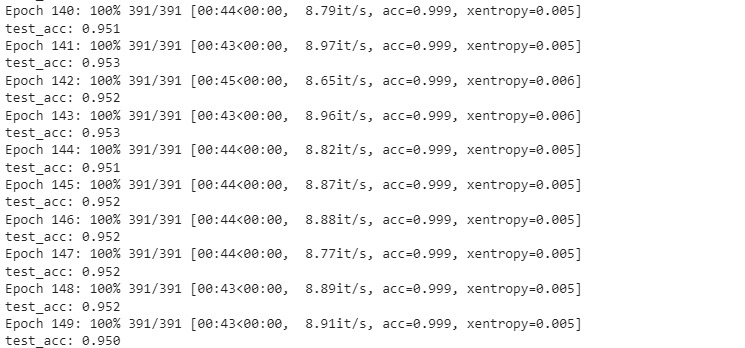
Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.RandomHorizontalFlip())

Model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_intensity\_shapes --shape triangle --length 16 --epochs 150

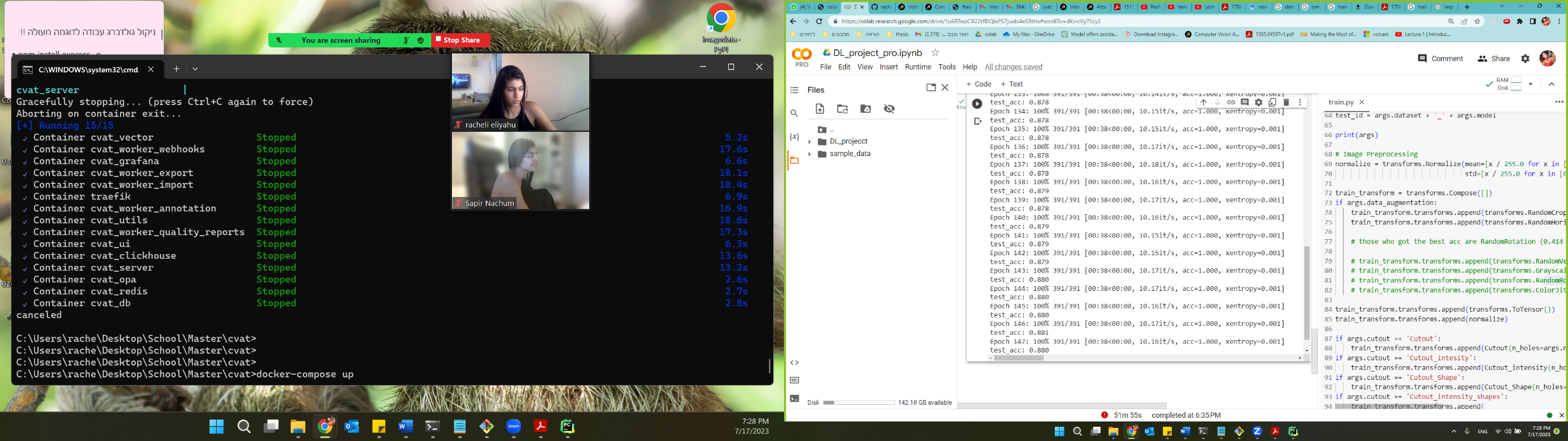


Epoch size: 150

Data augmentation taken: no data augmentation

Model:

!python train.py --dataset cifar10 --model resnet18 --epochs 150



Epoch size: 150

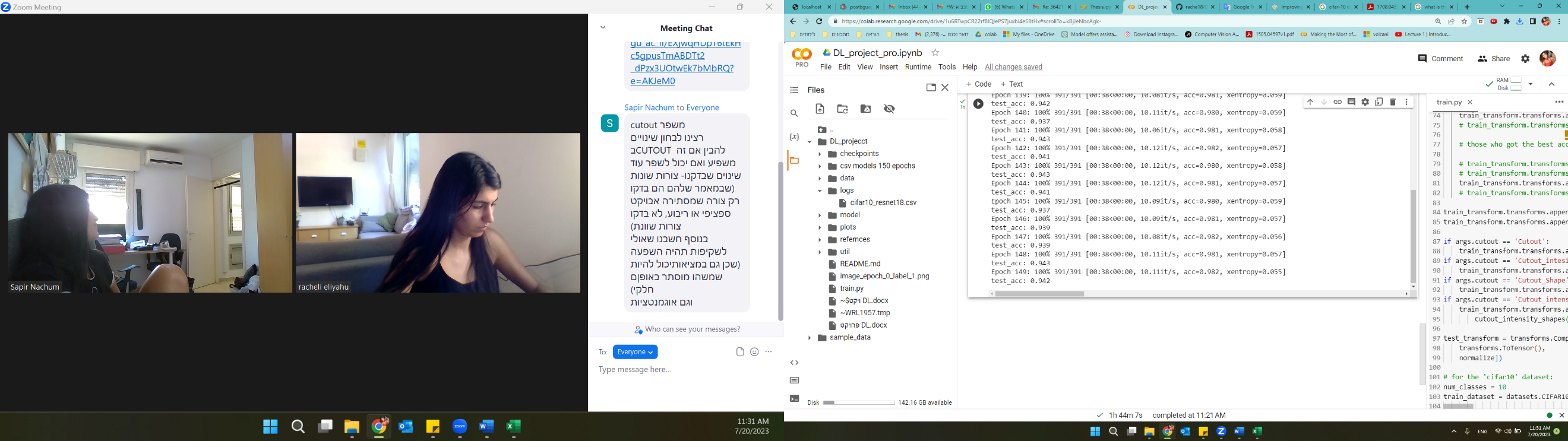
Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms. RandomRotation(10))

model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape square --length 16 --epochs 150



Epoch size: 150

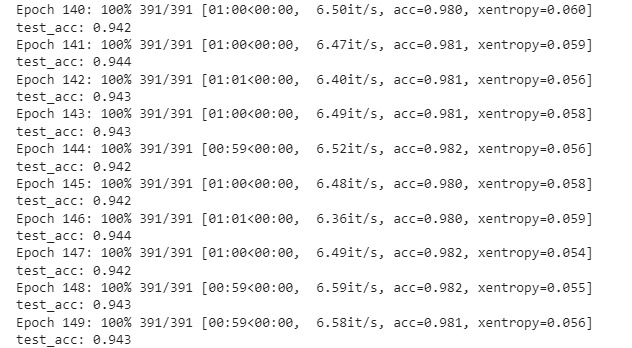
Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1))

model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape square --length 16 --epochs 150



Epoch size: 150

Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1))

train\_transform.transforms.append(transforms.RandomRotation(10))

model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape square --length 16 --epochs 150

A screenshot of a computer

Description automatically generated

Epoch size: 150

Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomCrop(32, padding=4))

    train\_transform.transforms.append(transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1))

train\_transform.transforms.append(transforms.RandomHorizontalFlip())

model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape square --length 16 --epochs 150

Epoch size: 150

Data augmentation taken:

    train\_transform.transforms.append(transforms.RandomRotation(10))

model:

!python train.py --dataset cifar10 --data\_augmentation --model resnet18 --cutout Cutout\_Shape --shape square --length 16 --epochs 150

Details of proposal

**Residual Network**

Residual networks (ResNets) operate by recasting each layer as a learning process that aims to acquire a residual function with respect to its input, thereby deviating from conventional methods that learn unreferenced functions. This residual function represents the discrepancy between the desired output and the input provided to that specific layer. By explicitly modeling and learning this residual function, ResNets achieve a more effective means of adjusting the input at each layer, thereby facilitating the training of deeper networks. The core principle behind ResNets involves the introduction of shortcut connections, which serve to reintegrate the residual function into the input of the same layer. This process establishes a bypass mechanism that circumvents the layer and allows the residual information to flow directly through the network. Such shortcut connections significantly mitigate the vanishing gradient issue, a challenge often encountered when training traditional deep neural networks. The utilization of these shortcut connections empowers ResNets to overcome the limitations associated with depth, enabling the development of considerably deeper networks. Moreover, the presence of these connections expedites the propagation of gradients throughout the network during the learning process, resulting in enhanced training efficiency and convergence. In summary, the architectural innovation of residual networks, which incorporates shortcut connections to model and learn residual functions, facilitates the training of deep networks, while simultaneously ensuring seamless gradient flow. This fundamental advancement has yielded substantial improvements in the performance and training capabilities of deep neural networks, marking a significant milestone in the field of deep learning research. Citation Deep Residual Learning for Image Recognition

A screenshot of a computer screen

Description automatically generated

Figure 1: Residual learning: a building block.

"The study conducted in Author(s), "Title of the Paper," Journal/Conference Name, Year has revealed a noteworthy observation in which the incorporation of randomly applied square Cutout to images manifests a progressive enhancement in the ResNets model's performance. This paper aims to explain our investigative undertaking, wherein we sought to investigate the influence of modifying the shape of the Cutout region on the resultant outcomes. Additionally, we conducted thorough assessments by employing diverse shapes, varying transparency percentages of the Cutout, and employing distinct data augmentation methodologies to comprehensively explore the potential impact on the model's efficacy

**Cutout shapes**

Cutout is a training technique that operates by randomly masking square regions of input data during the training process. This strategic masking compels the model to develop more resilient and diverse features by discouraging overreliance on any specific set of features. Consequently, the model becomes proficient at generalizing previously unseen data, leading to enhanced performance when evaluated on test sets. Notably, the simplicity and computational efficiency of cutout render it an accessible and seamless approach that can be seamlessly integrated with other data augmentation and regularization methodologies.

In our quest to identify the ideal shape for cutout, we conducted a comprehensive investigation encompassing three distinct shapes: rectangle, triangle, and circle. Through rigorous experimentation, we sought to discern the impact of these diverse shapes on the model's performance and generalization capabilities. By systematically varying the cutout shapes and meticulously analyzing their respective effects, we aimed to ascertain the most advantageous configuration to further augment the model's efficacy.

Rectangle: Square:

In our investigation of the cutout technique, we focused on employing a rectangular shape, characterized by a fixed side length which received by the user. We fixed the length to be 16 pixels. The application of cutout involves the random selection of a pixel coordinate within the image, serving as the center point for placing the cutout mask. By adopting this approach, the square region is dynamically positioned on the image during the training process, this approach allows for the possibility that parts of the cutout mask may extend beyond the image boundaries. This randomized placement of the 16-pixel square cutout introduces a dynamic and stochastic element to the training procedure, augmenting the model's adaptability and fostering the acquisition of robust features that are not solely reliant on the presence of this masked region. *Figure 2*

Triangle

For the triangle shape, the initial steps are like those used for the rectangle shape in the cutout implementation. We begin by identifying the center point within the image where the triangle will be placed. The fixed side length of the triangle is set at 16 pixels. Following this, we calculate the bounds of the triangle, ensuring that it remains within the image dimensions. Once we have determined the region for the triangle, a mask is applied to the image, selectively covering the corresponding pixels within the designated triangular area. *Figure 3*

Circle

The circle shape implementation involves the following steps: First, a random size between 8 to 16 pixels is chosen, and the circle's radius is determined as approximately half of the minimum dimension between the image's height and width based on the selected size. Then, pixel coordinates for the image's height and width are manually generated to create the image grid. Using a Boolean condition, a circular mask is formed by evaluating the distance of each pixel from the center point (x, y) and including pixels within the specified radius while excluding those outside it. Finally, the circular mask is applied to the image, effectively removing pixels within the circular region. This technique enriches the training data with circular regions, enhancing the model's ability to learn robust features and improve generalization. *Figure 4*

**Intensity**

In our investigation, we explored the effect of different transparency levels (intensity) on the cutout technique. The transparency value, denoted as "intensity," is randomly generated within the range of 0 to the maximum transparency value, which, in this case, is 0.5. A transparency value of 0 represents complete opacity, where the cutout region is entirely solid, covering the selected portion of the image without revealing any of the original content. Conversely, a transparency value of 0.5 means that the cutout region is 50% transparent, allowing 50% of the original content underneath to be visible through the masked region. By varying the intensity of the cutout, we aimed to examine how different transparency levels influence the model's generalization capabilities and its ability to learn robust features. The investigation allowed us to explore the trade-off between transparency levels and their impact on the model's performance when evaluated on previously unseen data.

**Data augmentation**

A blue car parked in front of a building

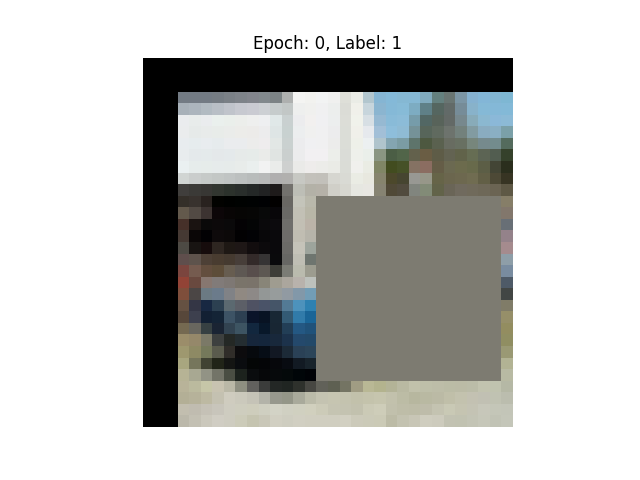
Description automatically generated****

Figure 2

Figure 3: Square CutOut with padding

Figure 4: Triangle CutOut with padding

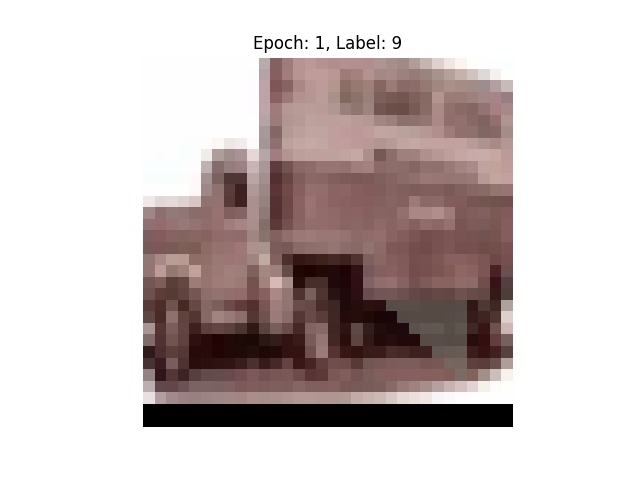
****

Figure 5: trnaperant triangle

**CIFAR-10 and Analysis**

Each model is implemented on the CIFAR-10 dataset that contains 60,000 32x32 color images in 10 different classes.

…All models were 150 epochs