**Assignment 4**

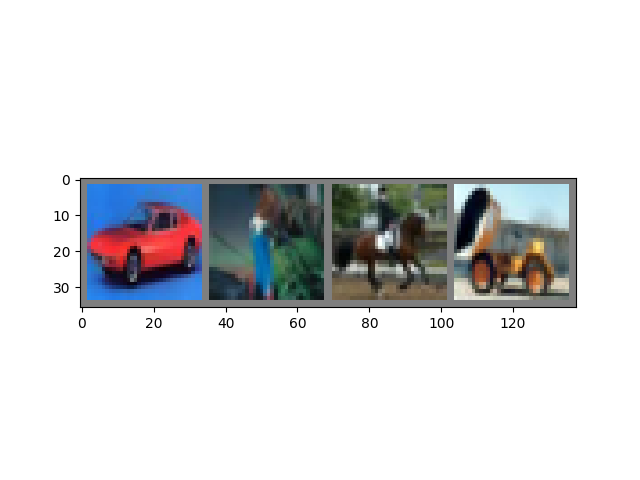
**Ofir Gilad & Max Brener**

**2 Tasks:**

**1. CIFAR-10 classification:**

*The tutorial results:*

Example images:



car bird horse truck

Train results:

[1, 2000] loss: 2.215  
[1, 4000] loss: 1.924  
[1, 6000] loss: 1.725  
[1, 8000] loss: 1.616  
[1, 10000] loss: 1.546  
[1, 12000] loss: 1.471  
[2, 2000] loss: 1.400  
[2, 4000] loss: 1.378  
[2, 6000] loss: 1.355  
[2, 8000] loss: 1.345  
[2, 10000] loss: 1.311  
[2, 12000] loss: 1.264  
Finished Training

Test results:

A row of images of boats

Description automatically generated

**Ground Truth:** cat, ship, ship, plane

**Predicted:** cat, ship, ship, plane

Accuracy of the network on the 10000 test images: 54.3 %

Accuracy for class: plane is 67.9 %  
Accuracy for class: car is 57.2 %  
Accuracy for class: bird is 47.1 %  
Accuracy for class: cat is 12.2 %  
Accuracy for class: deer is 51.1 %  
Accuracy for class: dog is 63.3 %  
Accuracy for class: frog is 51.4 %  
Accuracy for class: horse is 65.4 %  
Accuracy for class: ship is 64.0 %  
Accuracy for class: truck is 63.4 %

Explanation:

In this task we have repeated the same experiment that was done in the tutorial:

Firstly, we have downloaded the CIFAR-10 train and test datasets (which contains 60,000 RGB images of size 32x32 for 10 different classes) using **torchvision**, normalized the images, loaded them to train and test data loaders using **torch** (**batch size** was set to 4) and displayed 4 example images corresponded to the classes: “car”, “bird”, “horse”, “truck” (as seen in **Example images** section above).

Afterwords, we have defined a CNN with an architecture consisting of:

* Layer 1: **Conv2d** with: **in channels**: 3, **out channels**: 6, **kernel size**: 5 (**stride** and **padding** are with default values).   
  Followed by **ReLU** and **MaxPool2d** with: **kernel size**: 2, **stride**: 2.
* Layer 2: **Conv2d** with: **in channels**: 6, **out channels**: 16, **kernel size**: 5 (**stride** and **padding** are with default values).   
  Followed by **ReLU** and **MaxPool2d** with: **kernel size**: 2, **stride**: 2.
* Layer 3: **Linear** with: **in features**: 16\*5\*5, **out features**: 120.  
  Followed by **ReLU**.
* Layer 4: **Linear** with: **in features**: 120, **out features**: 84.  
  Followed by **ReLU**.
* Layer 3: **Linear** with: **in features**: 84, **out features**: 10  
  (The out features equal to 10 since we have 10 classes in CIFAR-10).

Then, we have defined **Cross Entropy Loss** criterion and **SGD** optimizer with: **learning rate**: 0.001, **momentum**: 0.9.   
And trained the network for 2 epochs.

The training process involved:

* Iterating over the training dataset in batches.
* Performing forward and backward passes to calculate the loss and updating the model's parameters accordingly.
* Printing the running loss each 2000 iterations (as seen in **Train result** section above).
* Saving the network weights after the training was completed.

Finally, we have loaded the model we have took a batch from the tested imaged, and displayed the ground truth labels compared to the model predicted labels (which for our luck were than same, and showed that the model might have trained well),  
and then we calculated the overall accuracy and per-class accuracy of the model and printed them (as seen in the **Test** **results** section above).

**Notice:** To make the code in this task more organized we used the **Trainer** class that helped us control which functions we want to use each time and is easy to maintain.

**2. Deconvolutional Model:**

*The task results:*

Accuracy of the network on the 10000 test images: 47.63 %

Examples of 3 images and their reconstruction:

A collage of images of two people

Description automatically generated A close-up of a boat

Description automatically generated A comparison of a picture of a boat

Description automatically generated

Explanation:

In this task we have adapted our network from Task 1 to include two deconvolutional layers after the two convolutional layer to get the same network as appears in **Figure 1** in the assignment:

A diagram of a computer

Description automatically generated

The deconvolutional layers architecture is as follows:

* Alternative Layer 3: **ConvTranspose2d** with: **in channels**: 16, **out channels**: 6, **kernel size**: 5, **stride**: 1, **padding**: 0, **output padding**: 0.  
  Applied after **MaxUnpool2d** with: **kernel size**: 2, **stride**: 2, and **ReLU**.
* Alternative Layer 4: **ConvTranspose2d** with: **in channels**: 6, **out channels**: 3, **kernel size**: 5, **stride**: 1, **padding**: 0, **output padding**: 0.  
  Applied after **MaxUnpool2d** with: **kernel size**: 2, **stride**: 2, and **ReLU**.

To use this network, we have defined 2 new flags:

* **enable\_deconvolutional\_model** - A flag inside the network from Task 1, that when set to “**True**”, the network will output both the **predicted label** and the **reconstructed image** in one call for “**forward**” for the given **input image** (While disabling the flag will change the network to the network from Task 1).
* **Reconstruction\_regularized** – A flag inside our Trainer class, that when set to “True”, it will tell the Trainer class to use the new network we mode we have defined in this class (with the deconvolutional layer) and use the following the new loss which is composed of the original **Cross Entropy Loss** and a new reconstruction term of **MSE Loss** but with **sum** over all the errors divided by the **number of channels** and the **batch size**.

We trained the new network for 2 epochs (as we did in Task 1) with the new loss we have defined and plotted 3 example images with their reconstruction (as seen in **The task results** section above).

From the plotted images, we can observe that while the network did not achieve highly accurate reconstructions, it did manage to capture and reconstruct the main objects in the input images.   
The reconstructed images resemble the original objects, despite some loss in detail and accuracy.

**3. Latent Representations Analysis:**

*Train image:*

A close up of a train image

Description automatically generated A close-up of a train layer

Description automatically generated A collage of images

Description automatically generated

*Test image:*

A blurry image of an elephant

Description automatically generated A test image layer of a test

Description automatically generated with medium confidence A test image of a layer

Description automatically generated with medium confidence

Explanation:

For the first convolutional layer, setting five channels to zero at a time, we observed varying contributions to the image's structure. Some channels retained edges and textures, indicating their role in capturing low-level features.

For the second convolutional layer, focusing on three channels, the reconstructions highlighted the main object of the image's class. This suggests these channels capture higher-level semantic features.

Overall, we saw the following things:

* The first layer channels preserved basic structural details, with each channel contributing differently.
* The second layer channels revealed class-specific patterns, indicating they capture more abstract and meaningful features.
* This demonstrates a hierarchy in the model's feature abstraction, with lower layers focusing on fundamental visual elements and higher layers on complex, class-specific details.