**Wet part report**

**Submitted by:**

|  |  |  |
| --- | --- | --- |
| Name | ID | Email |
| Husain Azaqy | 213382104 | Hssain.azaqy@campus.technion.ac.il |
| Tomer Danilin | 207933961 | tomerdanilin@campus.technion.ac.il |

1.4. Evaluation

1. Reasoning and solution overview

MNIST Autoencoder architecture for task 1.2.1:

Encoder:

MNIST img 28X28

Conv 14X14X32

BatchNorm 32

GELU

Conv 7X7X64

BatchNorm 64

14X14X32

GELU

Flatten vector 3136

Fully con

3136->128

Decoder:

Latent space 128

dropout

Fully con

128->3136

Unflatten

7X7X64

transConv 14X14X32

BatchNorm 32

14X14X32

GELU

transConv 28X28X1

tanh

Considerations:

* Few conv layers for compact model and better results - no vanishing gradients
* GELU layer for smooth transitions and handling negative inputs, better generalization by nonlinearity of the function, based on standard normal distribution which aligns well with probabilistic interpretations.
* Dropout layer for improved generalization and decrease overfitting
* Batchnorm for faster convergence, better regularization and gradient flow

Param choices and hyperparameters tuning:

* Network depth
* Hidden dimensions
* Kernel size
* Learning rate and scheduling
* Batch size
* Dropout probability

CIFAR10 Autoencoder architecture for task 1.2.1

Encoder:

CIFAR10 img 32X32X3

Conv 16X16X32

BatchNorm 32

GELU

Conv 8X8X64

BatchNorm 64

14X14X32

GELU

Conv 4X4X128

BatchNorm 128

GELU

Flatten vector 2048

Fully con

2048128

Decoder:

Latent space 128

Fully con

1282048

Unflatten Vector 4X4X128

TransConv 8X8X64

BatchNorm 64

14X14X32

GELU

TransConv 16X16X32

BatchNorm 32

GELU

TransConv 32X32X3

tanh

Considerations:

* Few conv layers for compact model and better results - no vanishing gradients
* GELU layer for smooth transitions and handling negative inputs, better generalization by nonlinearity of the function, based on standard normal distribution which aligns well with probabilistic interpretations.
* No Dropout layer to not lose meaningful representation information
* Batchnorm for faster convergence, better regularization and gradient flow

Param choices and hyperparameters tuning:

* Network depth
* Hidden dimensions
* Kernel size
* Learning rate and scheduling
* Batch size
* Dropout probability

Joint classifier architecture for task 1.21

Latent space 128

Fully con

128->1024

GELU

Fully con

1024128

GELU

BatchNorm 128

BatchNorm 1024

dropout

dropout

Fully con

12810

Considerations:

* Compact classifier with linear layers for extracting the features from the latent space and then classifying it to the 10 labels
* GELU layer for smooth transitions and handling negative inputs, better generalization by nonlinearity of the function, based on standard normal distribution which aligns well with probabilistic interpretations.
* Dropouts for better generalization of the classifier
* Batchnorm for faster convergence, better regularization and gradient flow

Param choices and hyperparameters tuning:

* Network depth
* Hidden dimensions
* Kernel size
* Learning rate and scheduling
* Batch size
* Dropout probability

MNIST architecture for task 1.2.2:

Encoder architecture is the same as in previous part however now the there is no decoder, and the training process doesn’t use reconstruction error but is trained on minimizing classification error jointly with the classifier. Note that the classifier architecture is also the same as in previous part.

Considerations:

* We remained with the previous encoder and classifier because of their relatively fast convergence and good results.
* We used a modified scheduler which decreases learning rate when achieving a plateau on the validation set we also decrease the learning rate in steps on a factor of 8 – as it is a binary shift of three and is useful for numerical stability.

Param choices and hyperparameters tuning:

* Network depth
* Hidden dimensions
* Kernel size
* Learning rate and scheduling
* Batch size
* Dropout probability

CIFAR10 architecture for task 1.2.2:

Encoder architecture is the same as in previous part however now the there is no decoder, and the training process doesn’t use reconstruction error but is trained on minimizing classification error jointly with the classifier. Note that the classifier architecture is also the same as in previous part.

Considerations:

* We remained with the previous encoder and classifier because of their relatively fast convergence and good results.
* We used a modified scheduler which decreases learning rate when achieving a plateau on the validation set, we also decrease the learning rate in steps on a factor of 8 – as it is a binary shift of three and is useful for numerical stability.
* We used weight initialization for faster network convergence.

Param choices and hyperparameters tuning:

* Network depth
* Hidden dimensions
* Kernel size
* Learning rate and scheduling
* Batch size
* Dropout probability

MNIST and CIFAR10 architecture for task 1.2.3:

In this section we used constructive learning specifically with the NT-Xent loss function as presented in simCLR.

**The loss function**

The NT-Xent loss is designed to pull the embeddings of positive pairs closer together in the embedding space while pushing embeddings of all other pairs (negative pairs) further apart. NT-Xent leverages the structure of contrastive learning, the model learns robust, generalizable representations without needing labeled data.

**Temperature consideration**

We also chose a balanced temperature parameter T in the NT-Xent loss,

Small temperature makes the softmax distribution peak sharply for the nearest vector. The model is forced to “choose” the positive pair with high confidence, penalizing even small deviations harshly.

Large temperature flattens the distribution, spreading probability more evenly across all pairs. To large temperature can harm distinction between pairs and reduce test accuracy.

**Data augmentation**

Data augmentation is essential in SimCLR. Its used to create variability and robustness needed for the model to learn meaningful, generalizable representations, especially in self-supervised settings like contrastive learning. The goal is to train a model to recognize that two augmented versions of the same image are fundamentally the same, while distinguishing them from augmentations of different images, data augmentation provides the “different views” of the same data point that fuel this contrastive learning process. The main benefits of data augmentation are:

* Creating positive pairs and using NT-Xent loss to make them similar.
* Forces invariance and learns representations that are robust to such changes.
* Increase effective data size.
* Sharpening feature extraction by differentiating between meaningful features of the images and not the augmentations

We used the following augmentations with varying probability:

* Random crop
* Random horizontal flip
* Random color jitter
* Random gaussian blur with random strength

**Training process**

In the training process we used the NTXentLoss an Adam optimizer and scheduling as we described in the other parts.

Param choices and hyperparameters tuning:

* Augmentation type and probability
* Learning rate and scheduling
* Temperature for NTXentLoss
* Adam optimizer parameters
* Dropout probability

1. Quantative results

1.2.1 Mnist

|  |  |  |
| --- | --- | --- |
| Training accuracy | Validation accuracy | Test accuracy |
|  |  |  |

Reconstruction error(mean absolute error):

1.2.1 CIFAR10

|  |  |  |
| --- | --- | --- |
| Training accuracy | Validation accuracy | Test accuracy |
|  |  |  |

Reconstruction error(mean absolute error):

1.2.2 Mnist

|  |  |  |
| --- | --- | --- |
| Training accuracy | Validation accuracy | Test accuracy |
|  |  |  |

1.2.2 CIFAR10

|  |  |  |
| --- | --- | --- |
| Training accuracy | Validation accuracy | Test accuracy |
|  |  |  |

1.2.3 Mnist

|  |
| --- |
| Test accuracy |
|  |

1.2.3 CIFAR10

|  |
| --- |
| Test accuracy |
|  |

Comparison of 1.2.3 results to the previous encoders:

1. Qualitive results

MNIST self supervised autoencoder

5 original images and their respective reconstruction:

CIFAR10 self supervised autoencoder

5 original images and their respective reconstruction:

1. Linear interpolation
2. t-SNE Projections