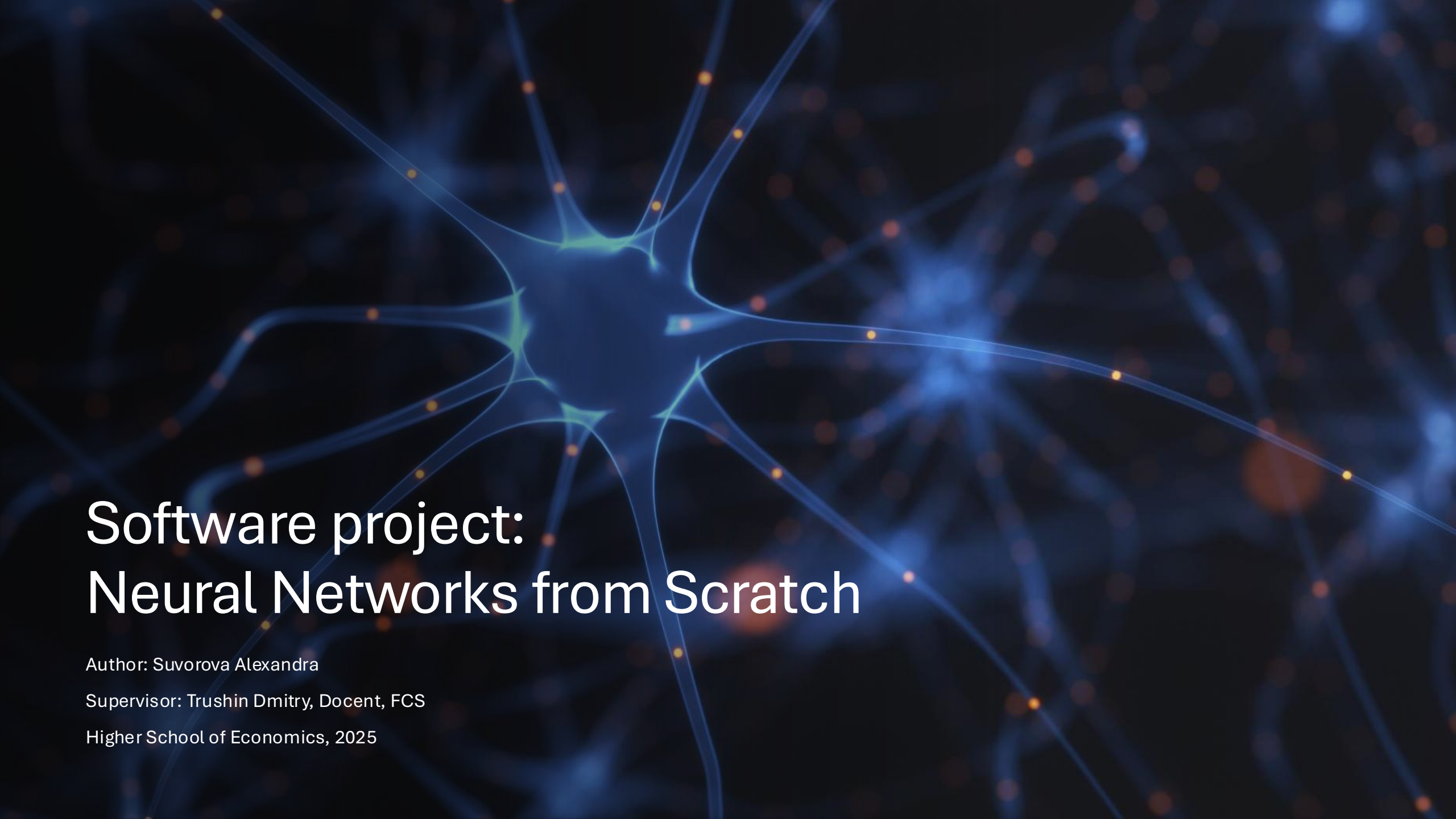


Software project:
Neural Networks from Scratch

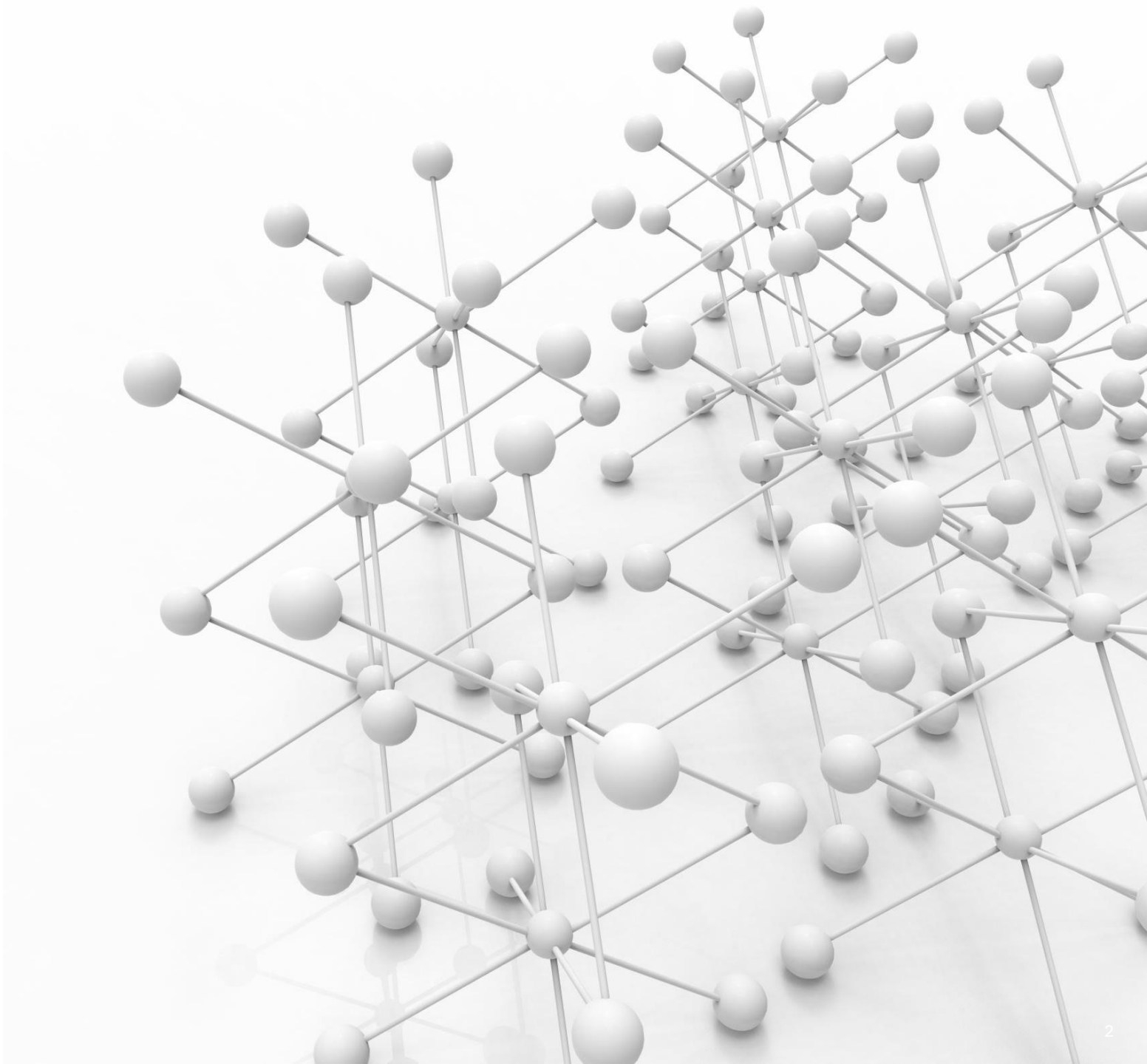
Author: Suvorova Alexandra

Supervisor: Trushin Dmitry, Docent, FCS

Higher School of Economics, 2025

Goal of the project

Implement a fully connected neural network from scratch as a library in C++



Main objectives

Implement a library with different activation and loss functions and optimizers

Design modular architecture

Train model using backpropagation and optimizers

Evaluate model performance

Create a clean interface

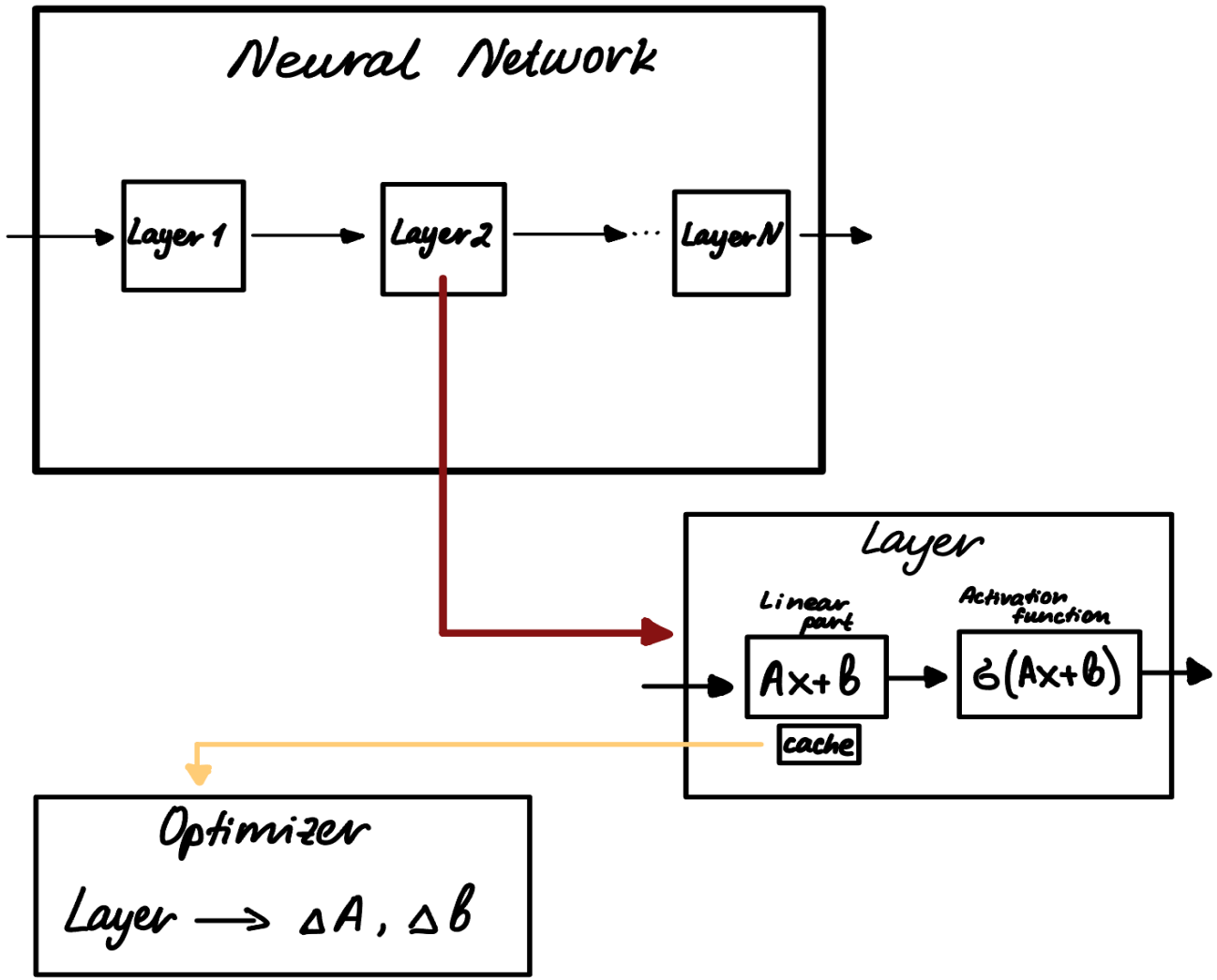
Why from Scratch?

Better understanding of core ML principles

Learn how backpropagation works under the hood

Practice with low-level implementation in C++

Net structure



Tech stack

- C++20
- Cmake version 3.10
- Git as a version control system
- LLVM code style
- Eigen and EigenRand for linear algebra computations

Features

Fully connected layers

- Linear + Non-linear

Polymorphism

- Type Erasure

Activation Function

- ReLU
- Sigmoid
- Tanh
- Identity
- Softmax

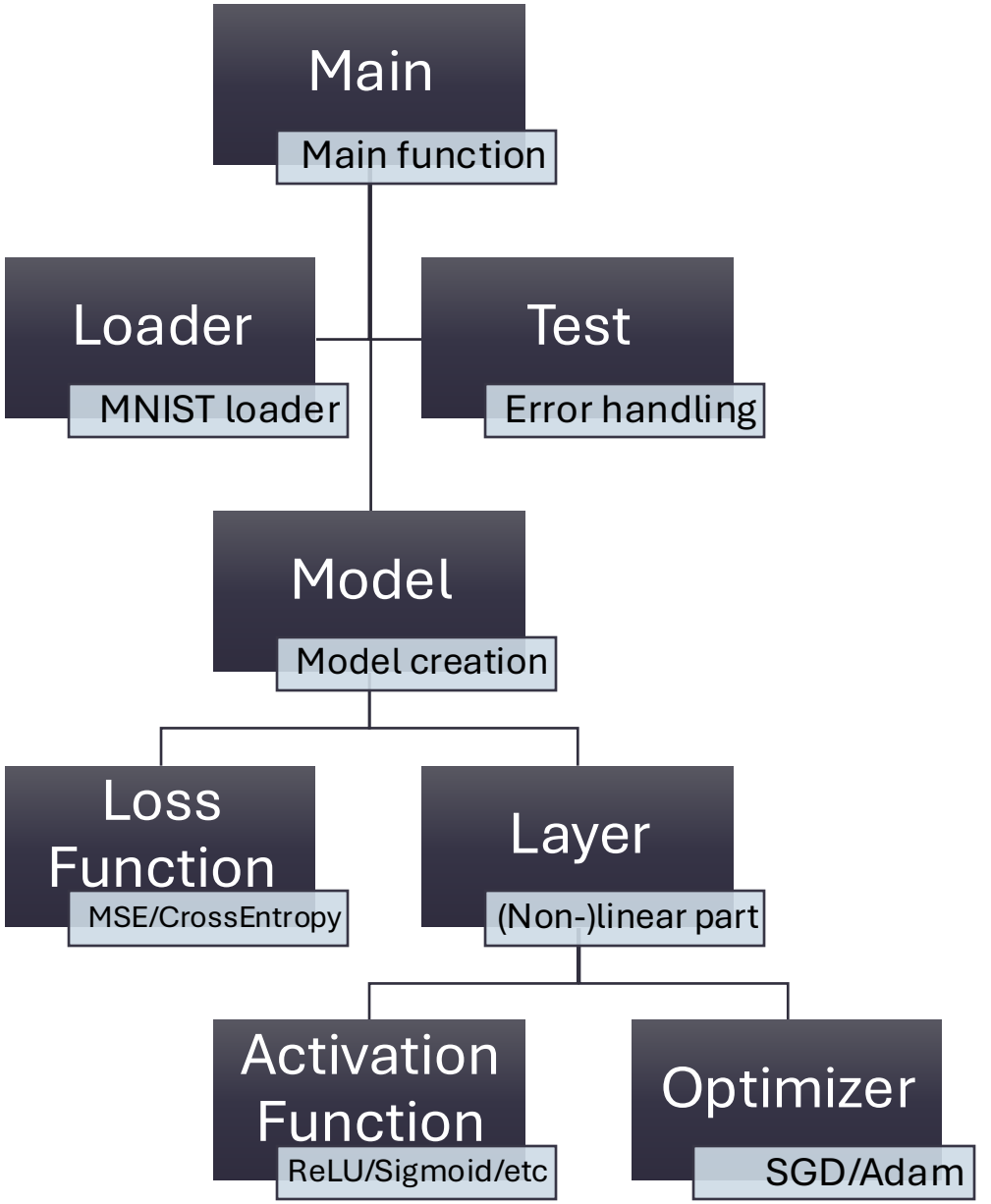
Loss Function

- MSE
- Cross Entropy

Optimizer

- SGD
- Adam

Code Structure and Modules



Code details

- Creation of a Model

```
Optimizer opt = Optimizer::Adam(0.001, 0.9, 0.999, 1e-8);
Model model({784, 128, 10}, {ActivationFunction::Type::ReLU,
                             ActivationFunction::Type::Identity});
```

- Launch of training

```
for (int i = 0; i < N; ++i) {
  model.trainStep(train_images[i], train_targets[i], LossFunction::mseGrad,
                  opt);
```

Testing

Correctness

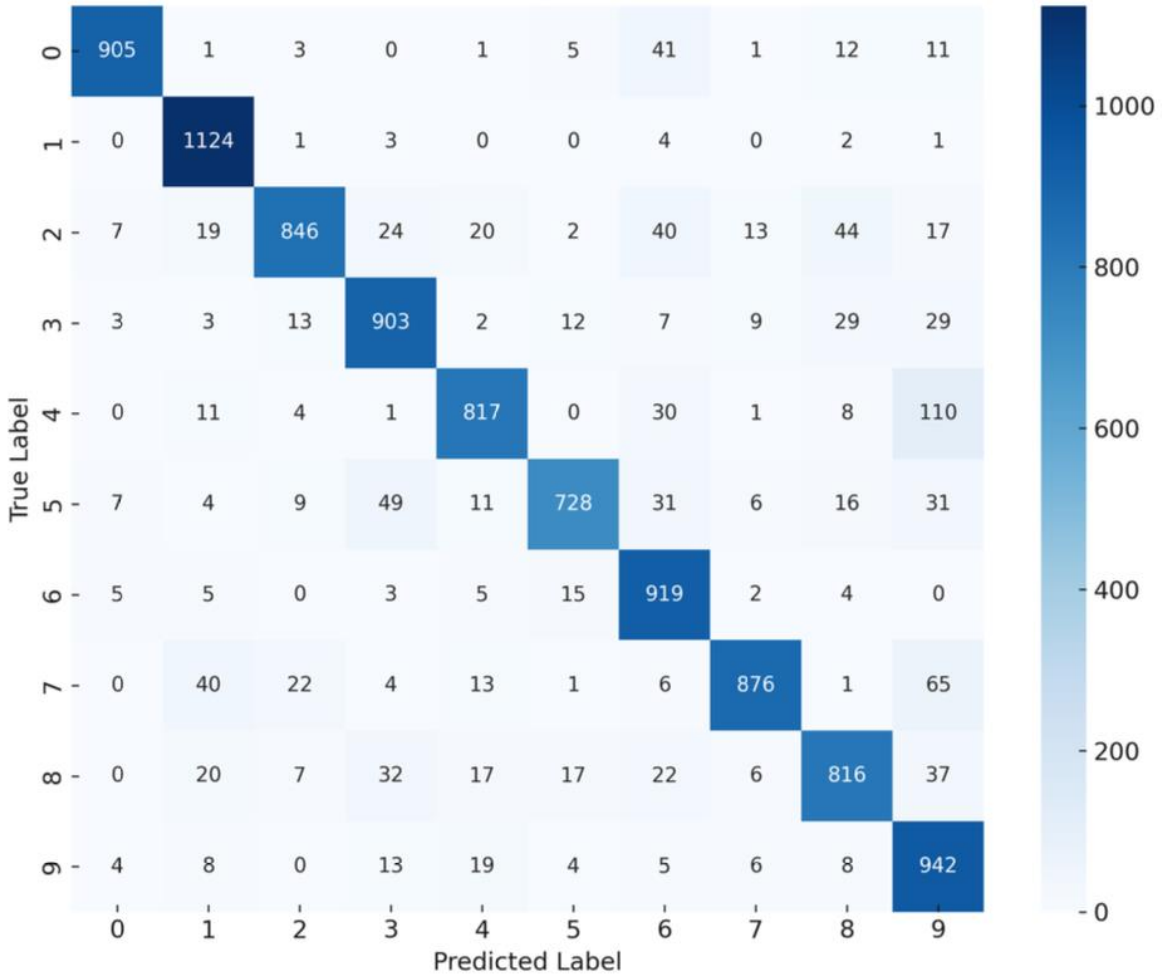
- Assert

Accuracy

- #successes/#all

MNIST dataset: handwritten digits

- 60,000 pictures train
- 10,000 pictures test



Confusion matrix

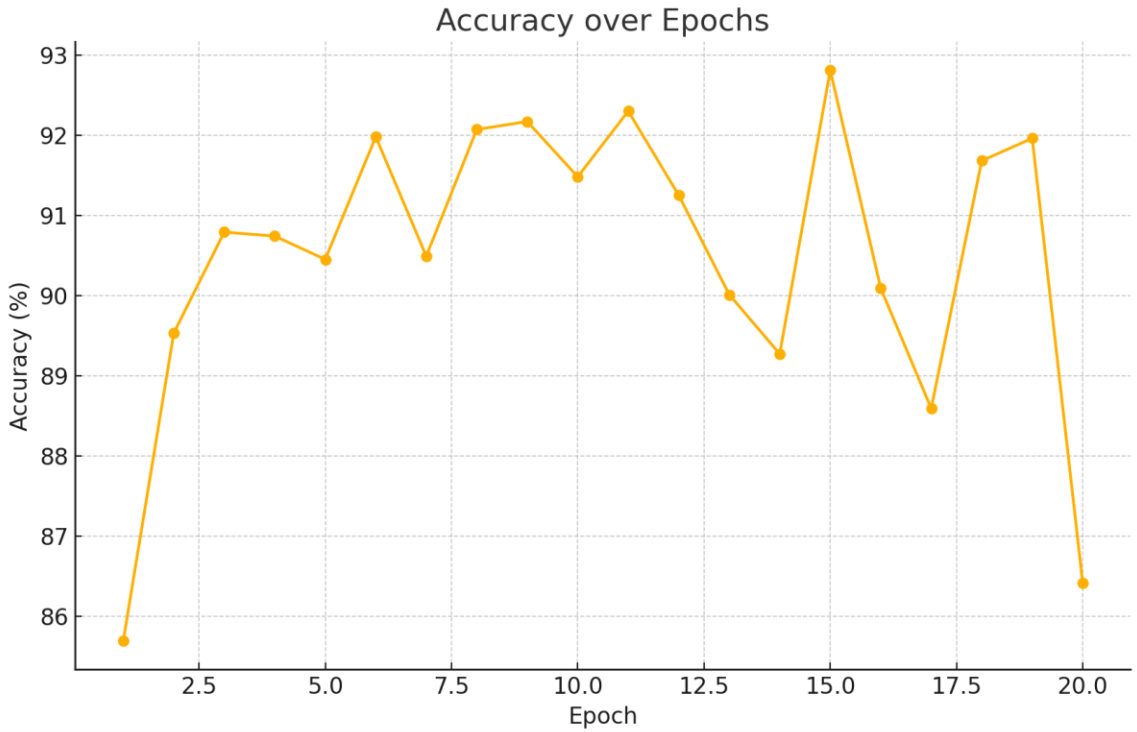
A[i][j] = # of samples of true class i that were predicted as class j

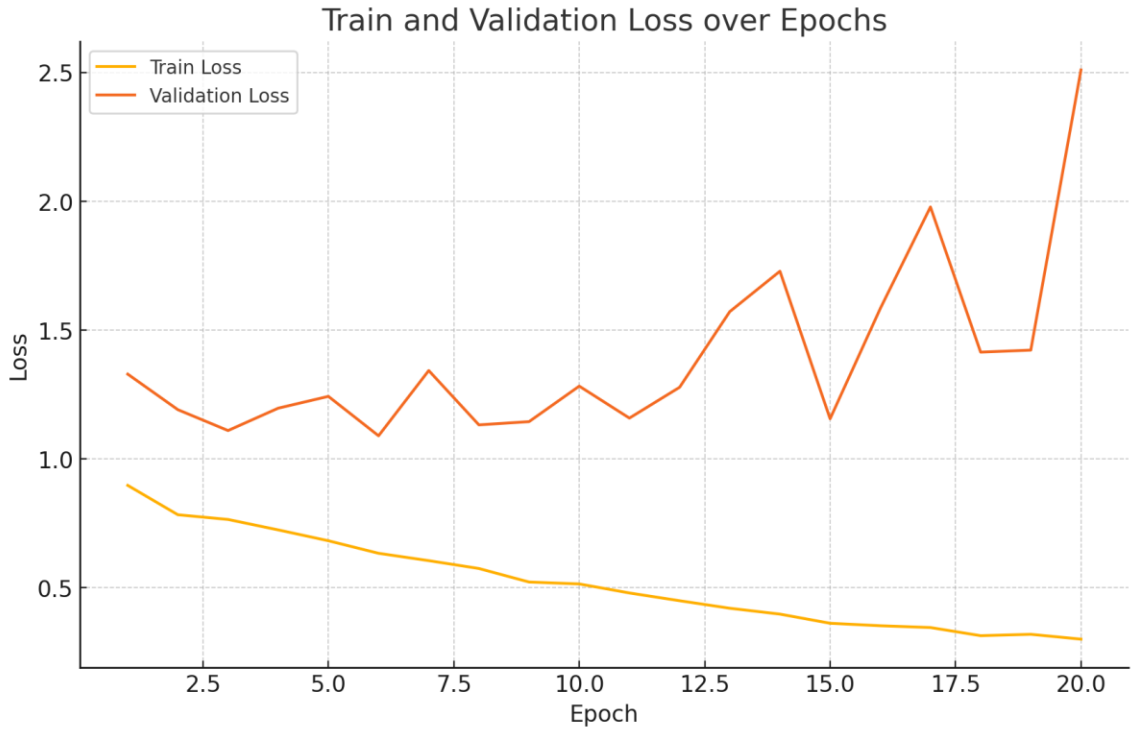
- One epoch of training
- 60,000 pictures – loss computation on each step

- 3 layers – 784-128-10
- ReLU-Identity
- MSE
- Adam

Comparison – 20 epochs

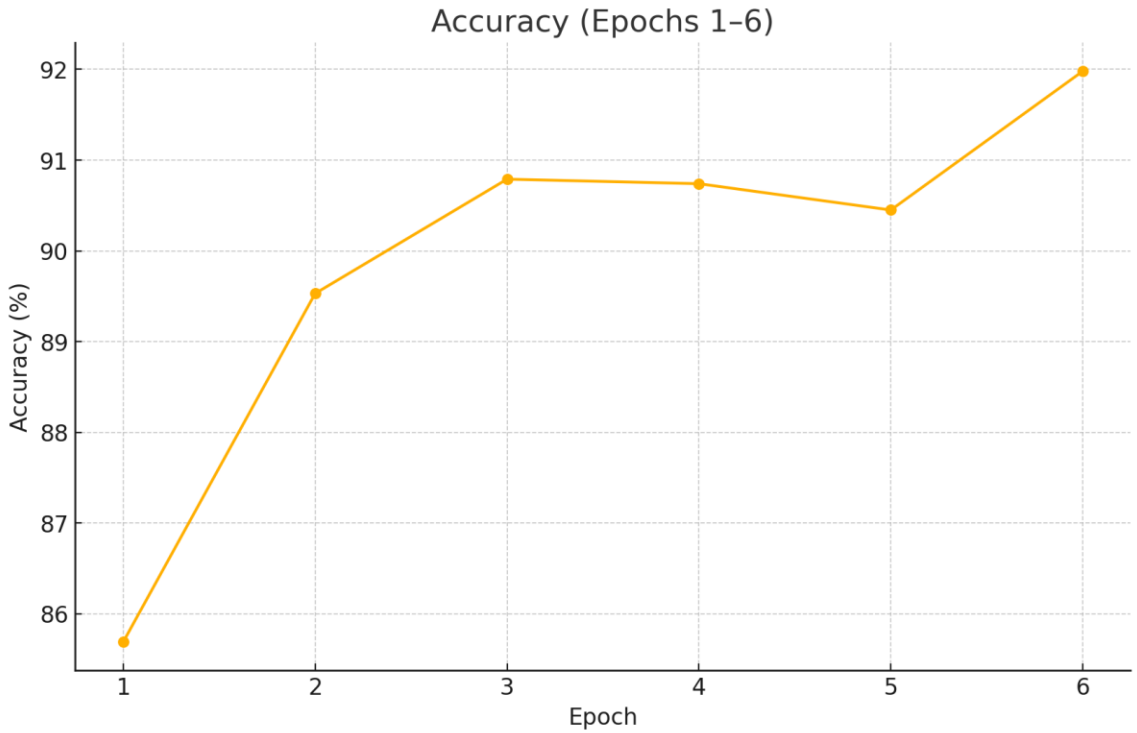
- 5 layers – 784-128-64-32-10
- ReLU-ReLU-ReLU-Softmax
- Cross Entropy
- Adam optimizer

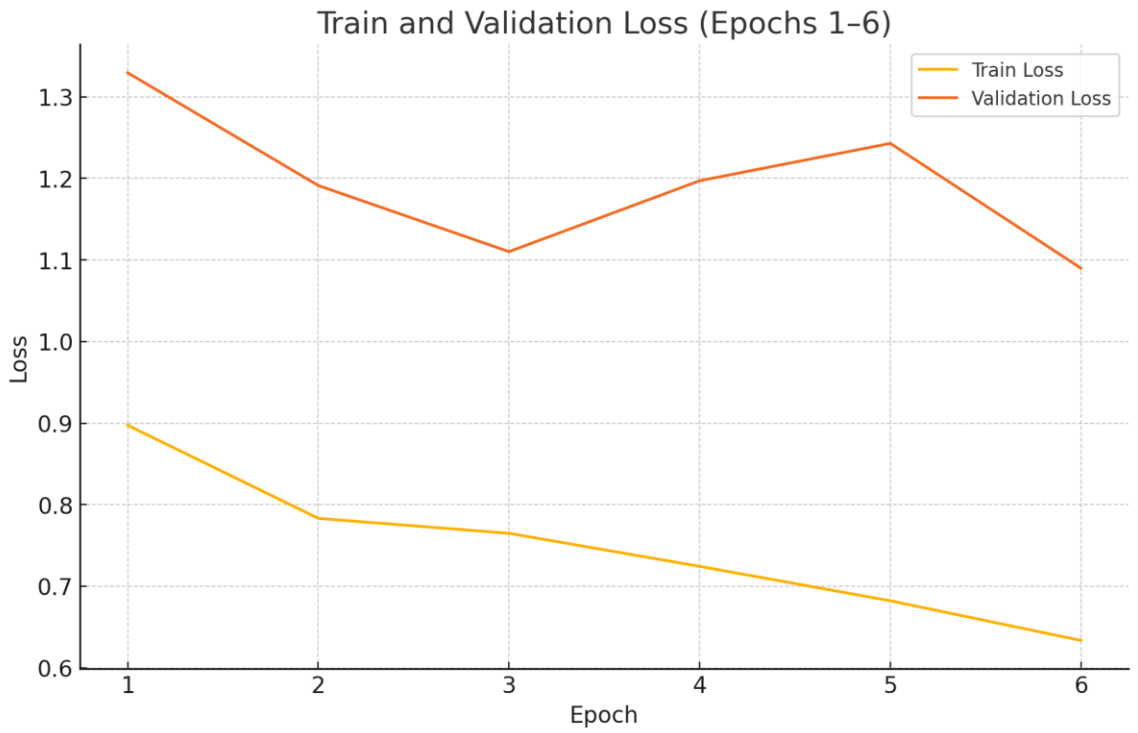




Comparison – 6 epochs

- 5 layers – 784-128-64-32-10
- ReLU-ReLU-ReLU-Softmax
- Cross Entropy
- Adam optimizer





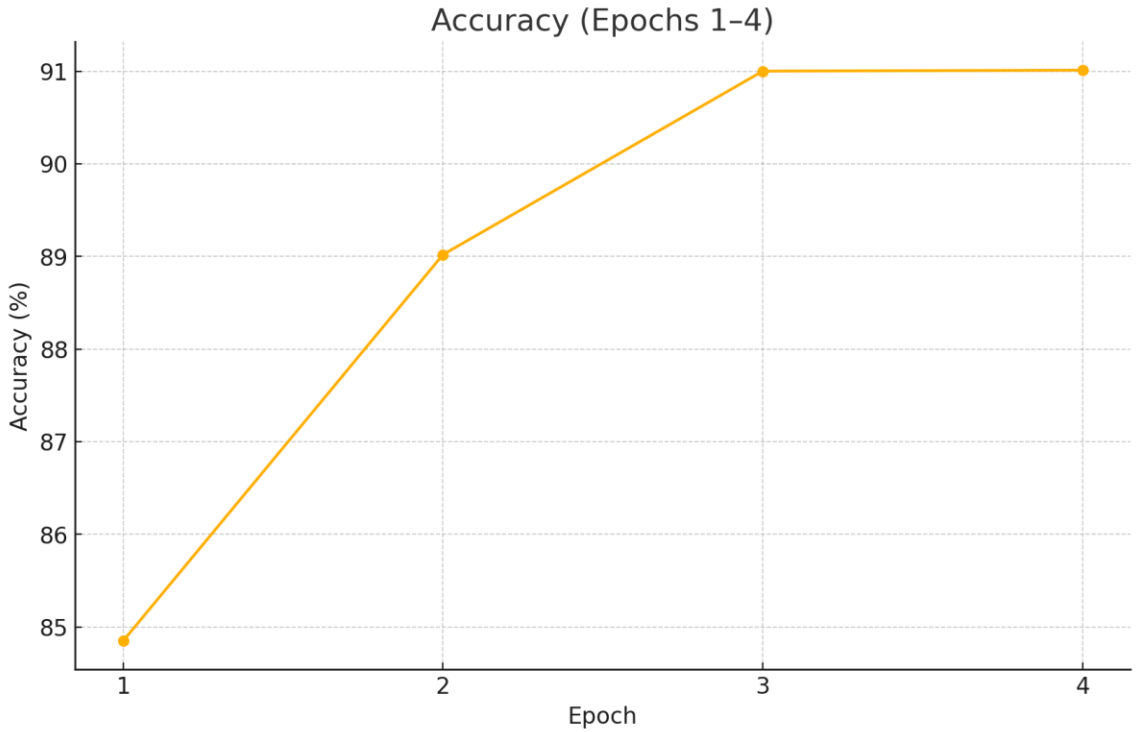
Remarkable result

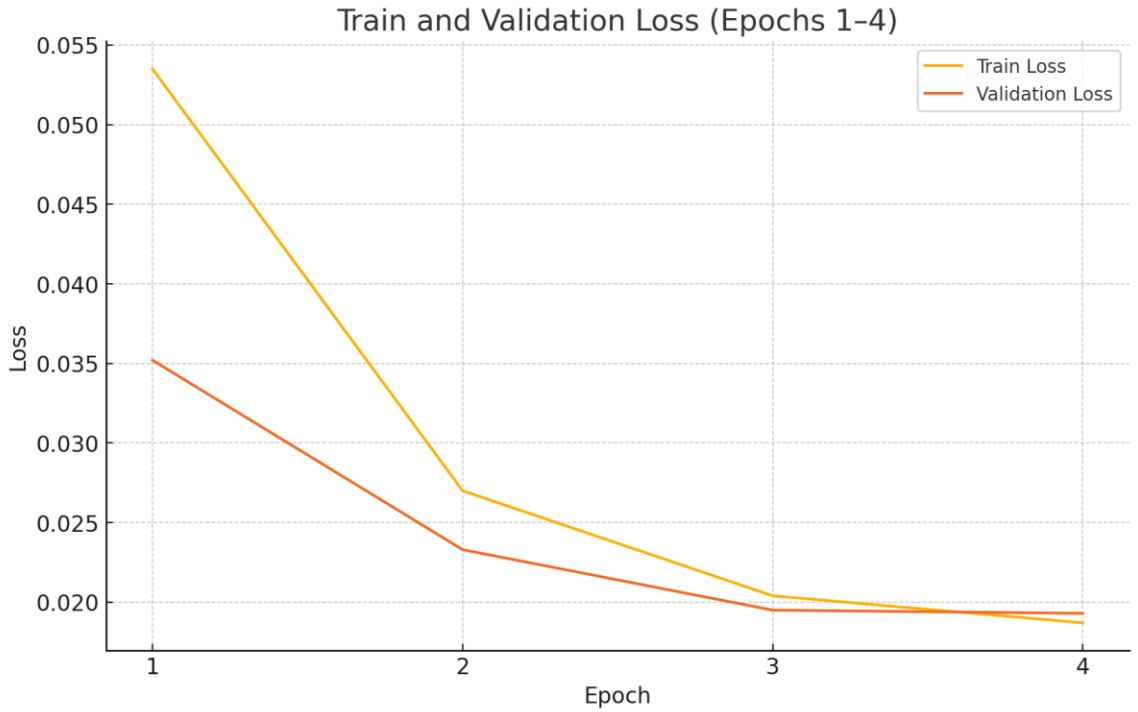
```
➜  build git:(dev) ✗ ./neural_net
[OK] All tests passed!
Select model architecture:
1. One hidden layer (ReLU + Identity)
2. Two hidden layers (ReLU + Sigmoid + Identity)
3. Three hidden layers (ReLU + ReLU + ReLU + Softmax)
Enter choice (1/2/3): 2
Enter number of training epochs: 4

=== Training model2 for 4 epoch(s) ===
[==============================] 100% (60000/60000) L:0.0535
Epoch 1 finished. Train Loss: 0.0535, Val Loss: 0.0352, Accuracy: 84.8500%
[==============================] 100% (60000/60000) L:0.0270
Epoch 2 finished. Train Loss: 0.0270, Val Loss: 0.0233, Accuracy: 89.0200%
[==============================] 100% (60000/60000) L:0.0204
Epoch 3 finished. Train Loss: 0.0204, Val Loss: 0.0195, Accuracy: 91.0000%
[==============================] 100% (60000/60000) L:0.0187
Epoch 4 finished. Train Loss: 0.0187, Val Loss: 0.0193, Accuracy: 91.0100%
```

Remarkable result

- 4 layers – 784-128-64-10
- ReLU-Sigmoid-Identity
- MSE
- Adam





Future work

01

Integrate GUI

02

Create a manual for students

03

Test on other datasets

Summary

Fully connected layers

- Forward and backward propagation
- Model creation

Activation functions:

- ReLU
- Tanh
- Identity
- Sigmoid
- Softmax

Loss functions:

- MSE
- Cross Entropy

Optimizers:

- SGD
- Adam

Tested on a MNIST dataset – 92% accuracy