МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Технический Университет Связи И Информатики (MTUCI)»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторная Работа 2

по дисциплине

«Машинное обучение»

Выполнил: студент 3 курса гр. БВТ2201
Ньяти Каелиле

Москва 2025 г

Содержание

1. Задание

Обучить MNIST. MNIST это набор картинок размер которых 1x28x28. Берите датасет в котором пиксель может быть вещественным числом от 0 до 1. Это задача классификации, на каждой картинке написана цифра, и нужно предктнуть что это за цифра.

2. Ход работы

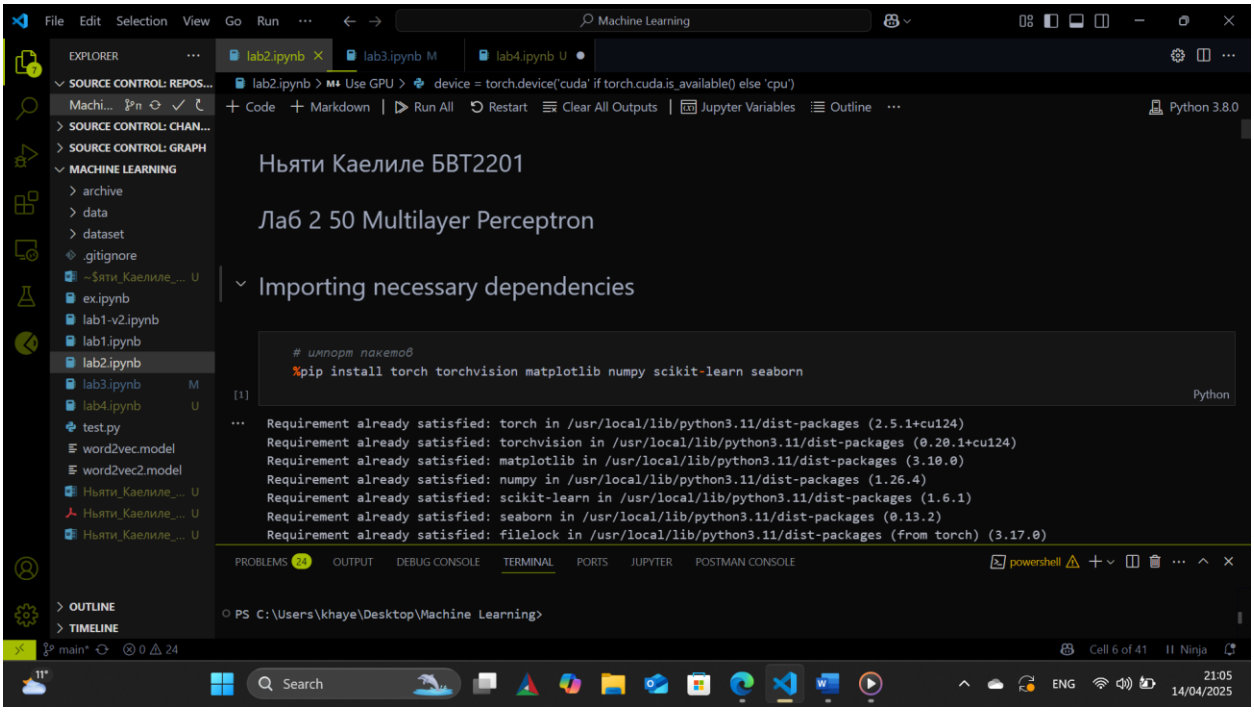
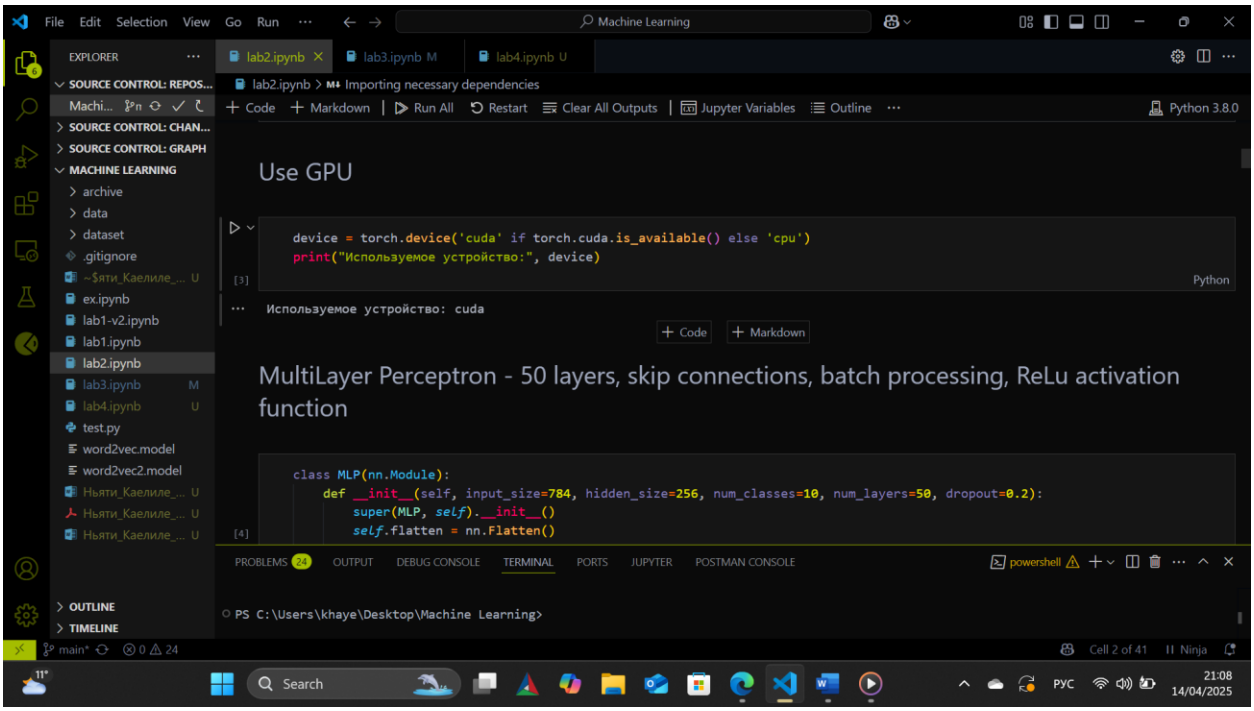
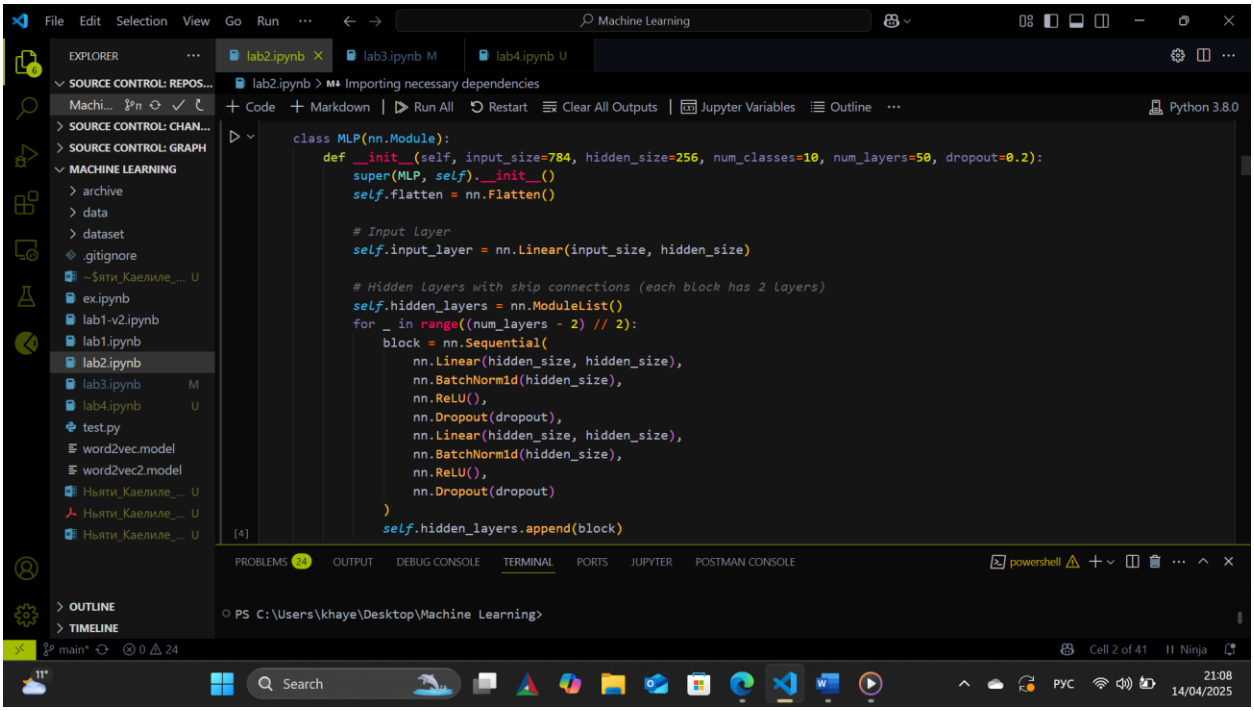
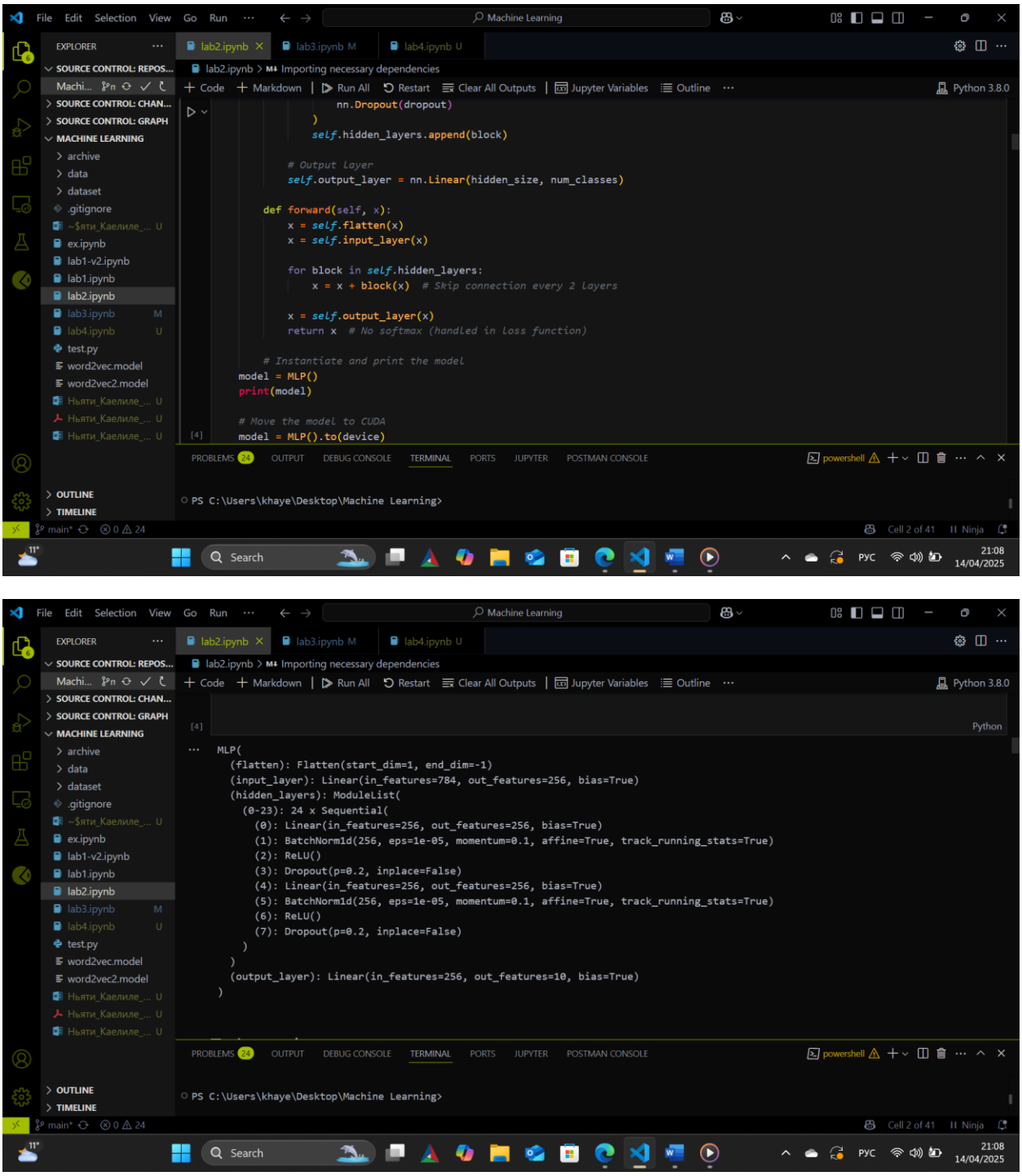


Рис 1. Библиотекы

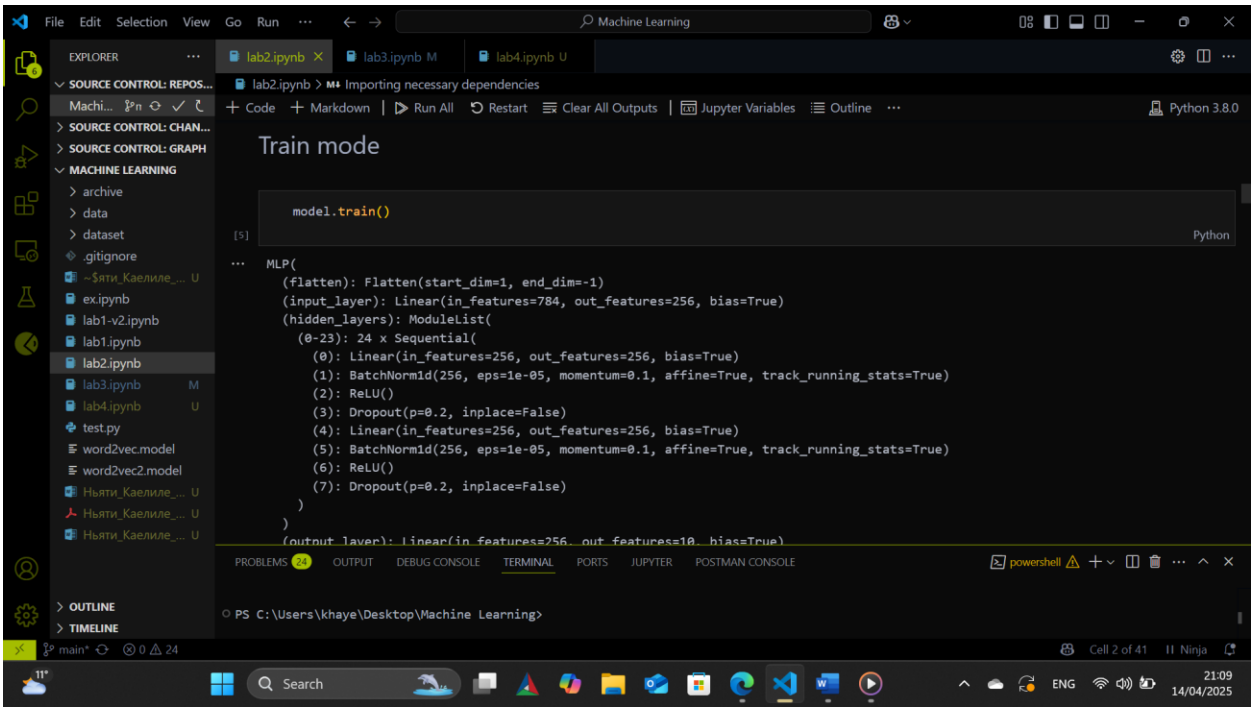

Use GPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print("Используемое устройство:", device)
Используемое устройство: cuda
MultiLayer Perceptron - 50 layers, skip connections, batch processing, ReLu activation function
class MLP(nn.Module):
def __init__(self, input_size=784, hidden_size=256, num_classes=10, num_layers=50, dropout=0.2):
super(MLP, self).__init__()
self.flatten = nn.Flatten()


class MLP(nn.Module):
def __init__(self, input_size=784, hidden_size=256, num_classes=10, num_layers=50, dropout=0.2):
super(MLP, self).__init__()
self.flatten = nn.Flatten()
# Input Layer
self.input_layer = nn.Linear(input_size, hidden_size)
# Hidden layers with skip connections (each block has 2 layers)
self.hidden_layers = nn.ModuleList()
for _ in range((num_layers - 2) // 2):
block = nn.Sequential(
nn.Linear(hidden_size, hidden_size),
nn.BatchNorm1d(hidden_size),
nn.ReLU(),
nn.Dropout(dropout),
nn.Linear(hidden_size, hidden_size),
nn.BatchNorm1d(hidden_size),
nn.ReLU(),
nn.Dropout(dropout)
)
self.hidden_layers.append(block)


File Edit Selection View Go Run
Machine Learning
EXPLORER
lab2.ipynb
lab3.ipynb M
lab4.ipynb U
Importing necessary dependencies
Code Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
Python 3.8.0
nn.Dropout(dropout)
self.hidden_layers.append(block)
# Output Layer
self.output_layer = nn.Linear(hidden_size, num_classes)
def forward(self, x):
x = self.flatten(x)
x = self.input_layer(x)
for block in self.hidden_layers:
x = x + block(x) # Skip connection every 2 layers
x = self.output_layer(x)
return x # No softmax (handled in loss function)
# Instantiate and print the model
model = MLP()
print(model)
# Move the model to CUDA
model = MLP().to(device)
[4]
PROBLEMS 24 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
OUTLINE
TIMELINE
main* 0 24
Cell 2 of 41 Ninja
Search
РУС
21:08
14/04/2025

File Edit Selection View Go Run
Machine Learning
EXPLORER
lab2.ipynb
lab3.ipynb M
lab4.ipynb U
Importing necessary dependencies
Code Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
Python 3.8.0
[4]
Python
MLP(
(flatten): Flatten(start_dim=1, end_dim=-1)
(input_layer): Linear(in_features=784, out_features=256, bias=True)
(hidden_layers): ModuleList(
(0-23): 24 x Sequential(
(0): Linear(in_features=256, out_features=256, bias=True)
(1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(2): ReLU()
(3): Dropout(p=0.2, inplace=False)
(4): Linear(in_features=256, out_features=256, bias=True)
(5): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(6): ReLU()
(7): Dropout(p=0.2, inplace=False)
)
)
(output_layer): Linear(in_features=256, out_features=10, bias=True)
)
PROBLEMS 24 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
OUTLINE
TIMELINE
main* 0 24
Cell 2 of 41 Ninja
Search
РУС
21:08
14/04/2025

Рис 2. MLP



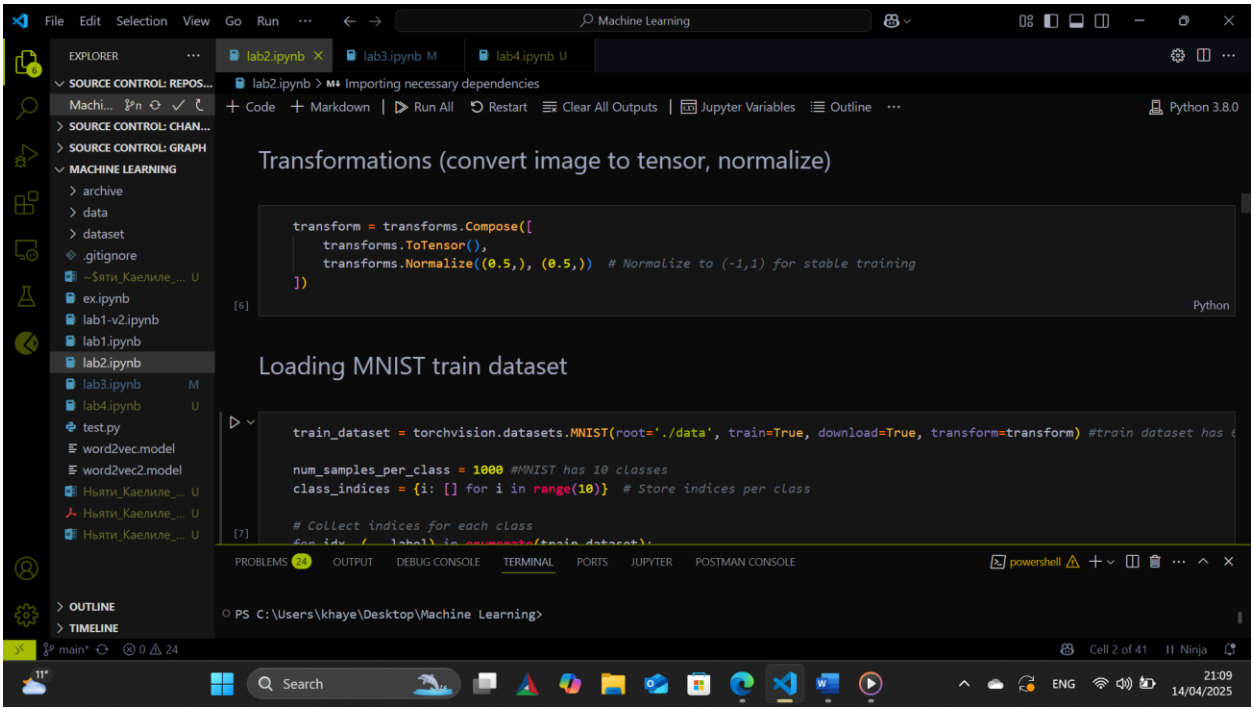
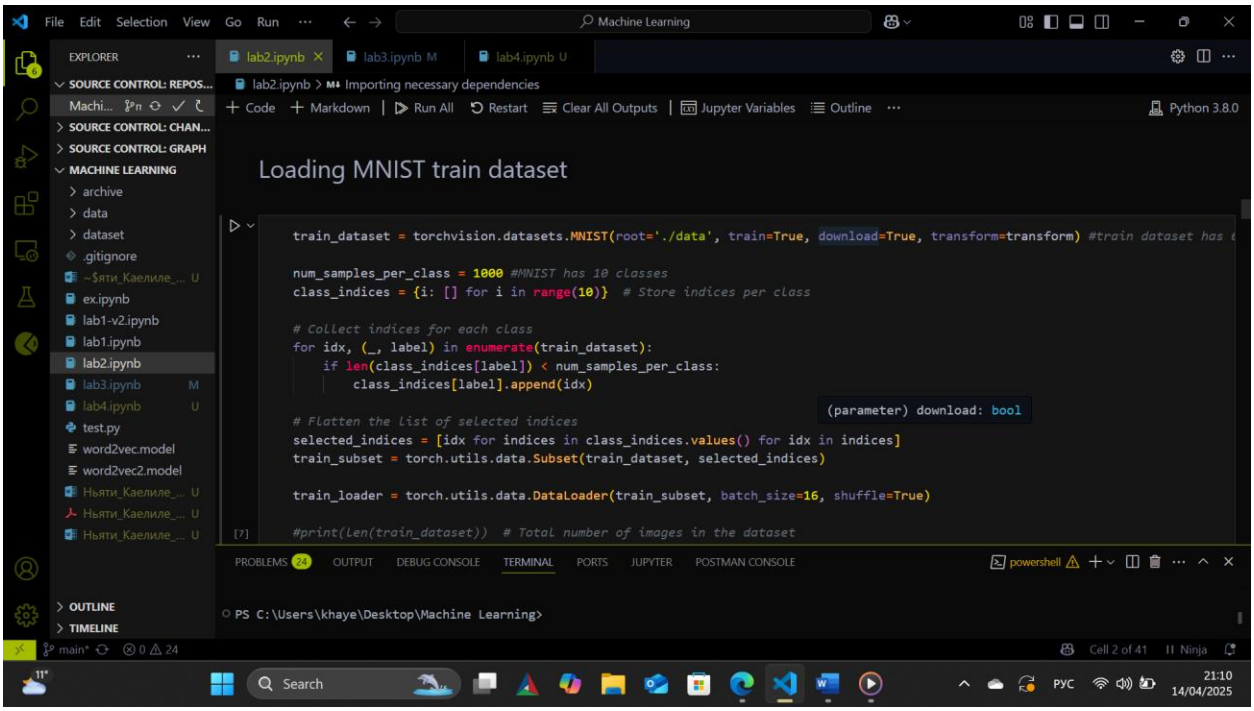


Рис 3. Трансформаций



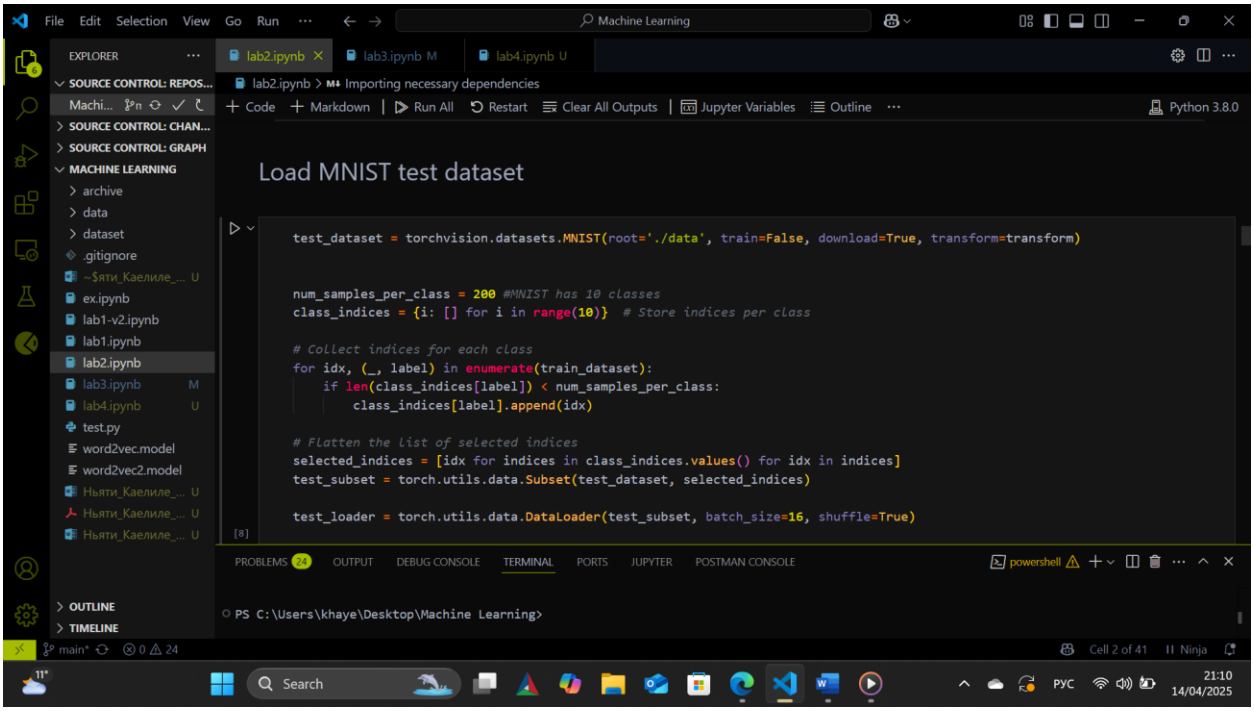


Рис 4. Дата лоудер

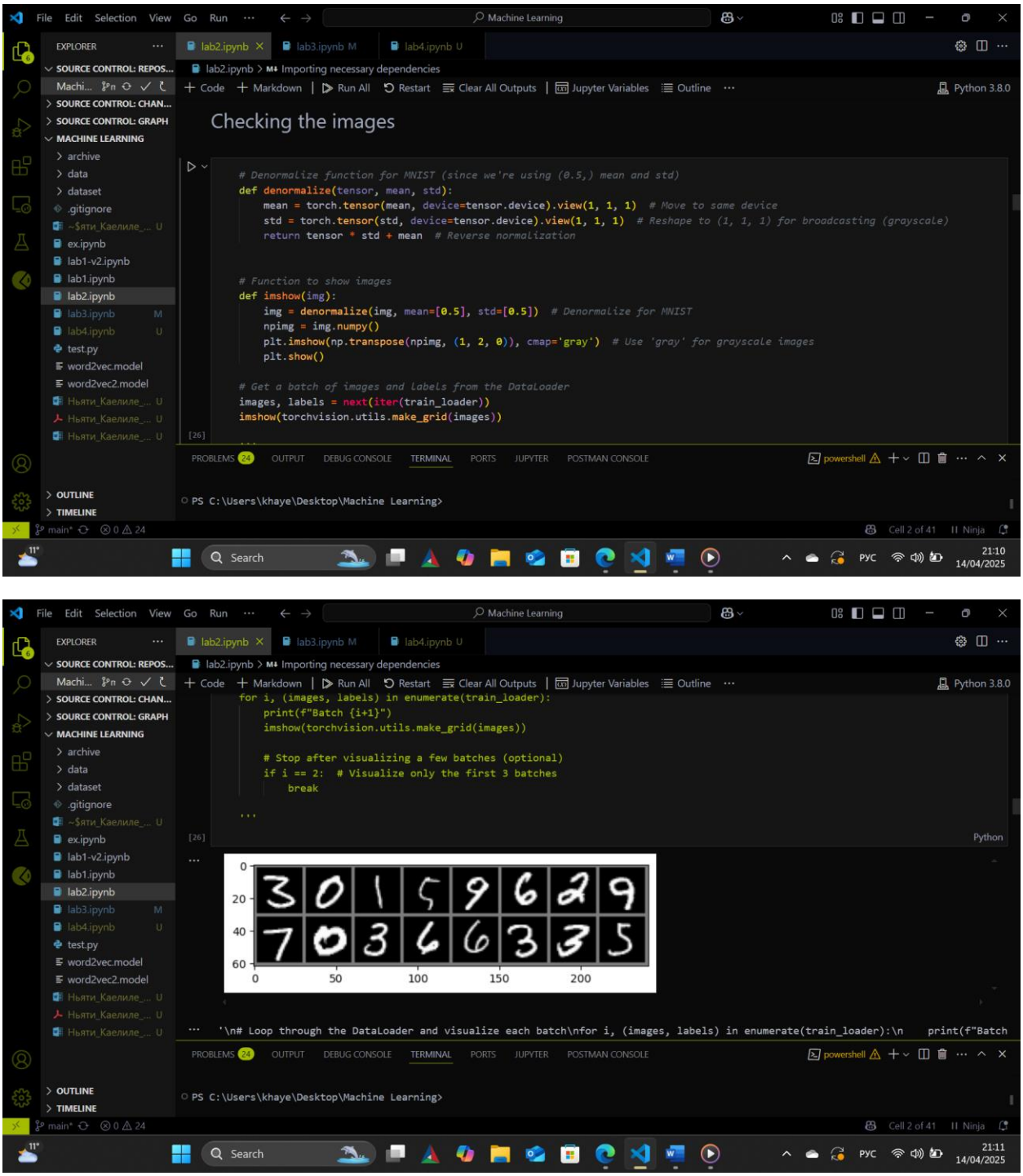


Рис 5. Изображение

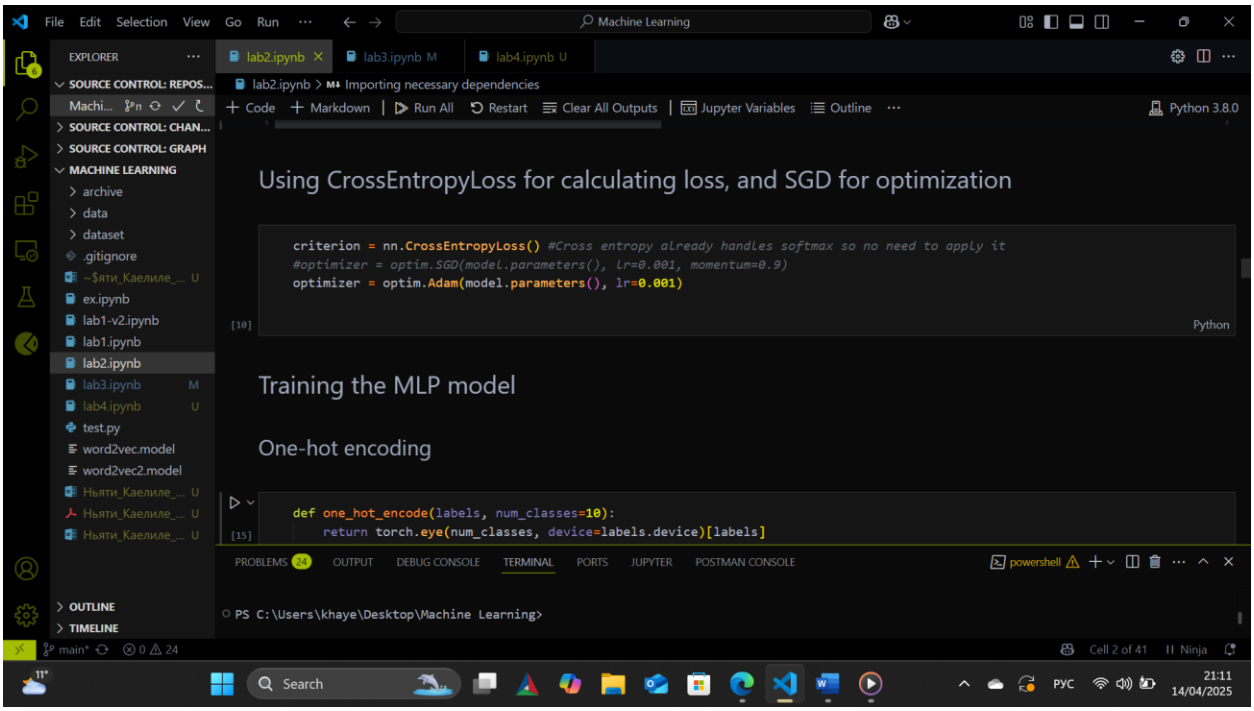


Рис 6. Loss function

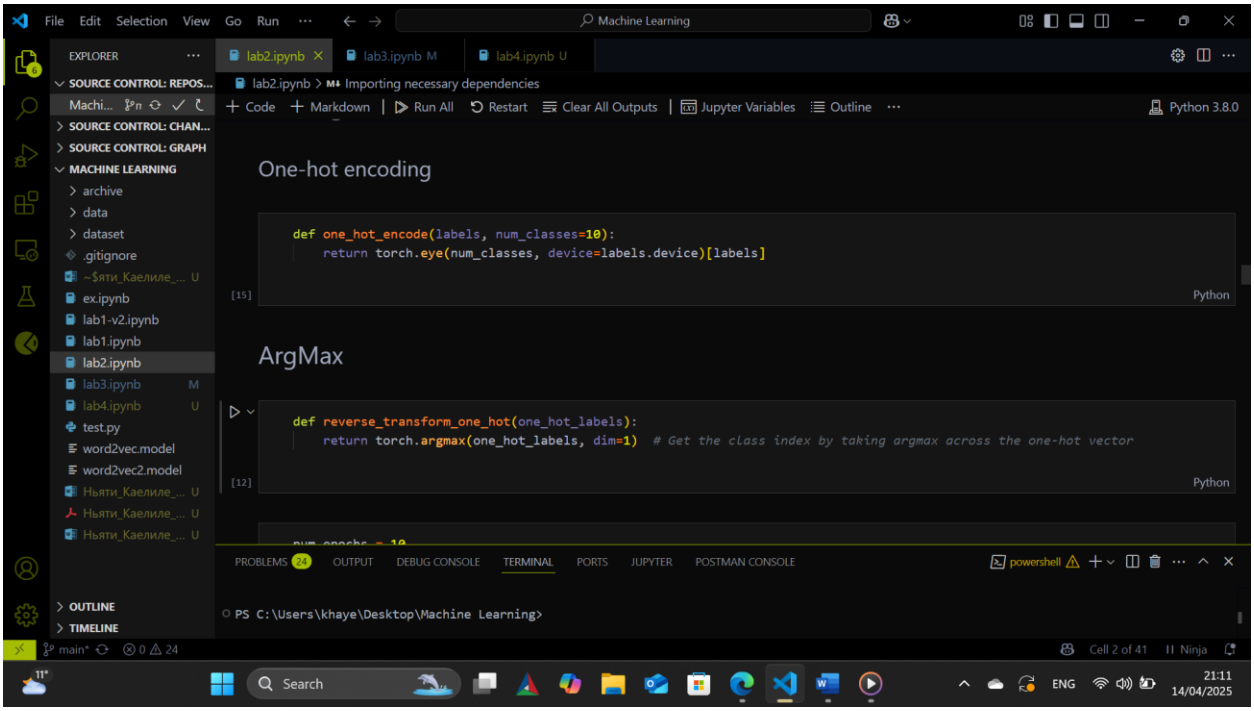
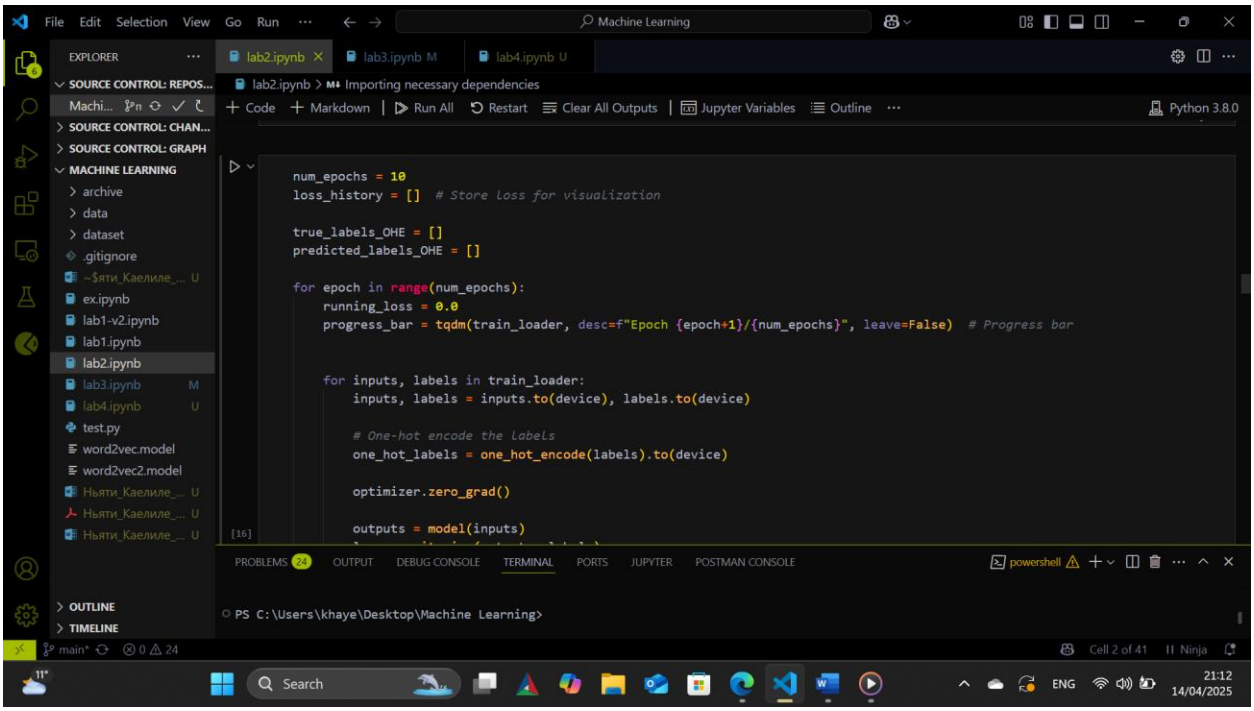
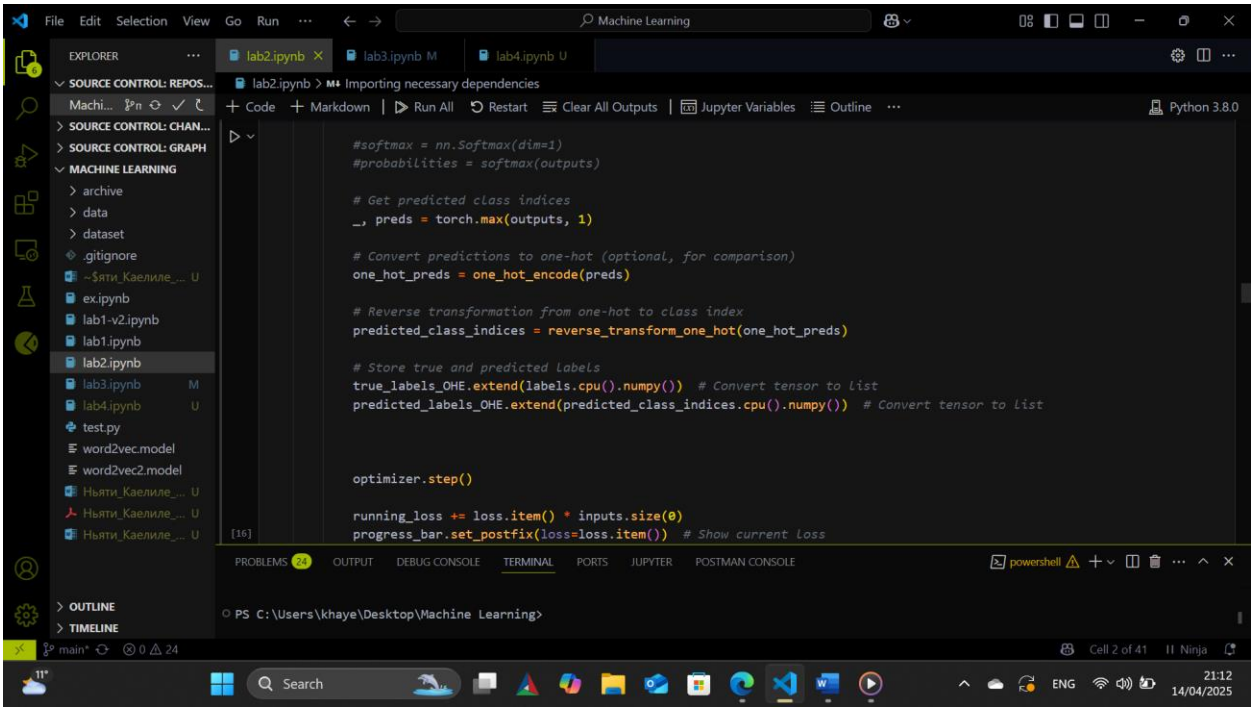
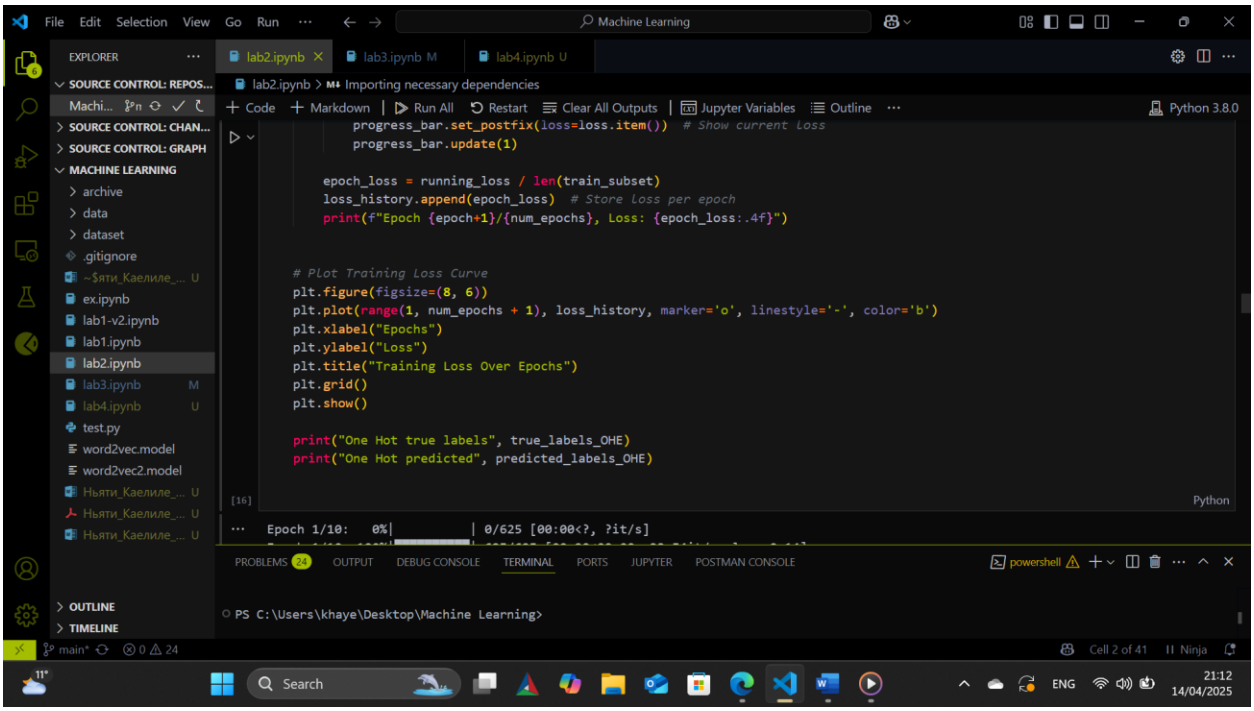
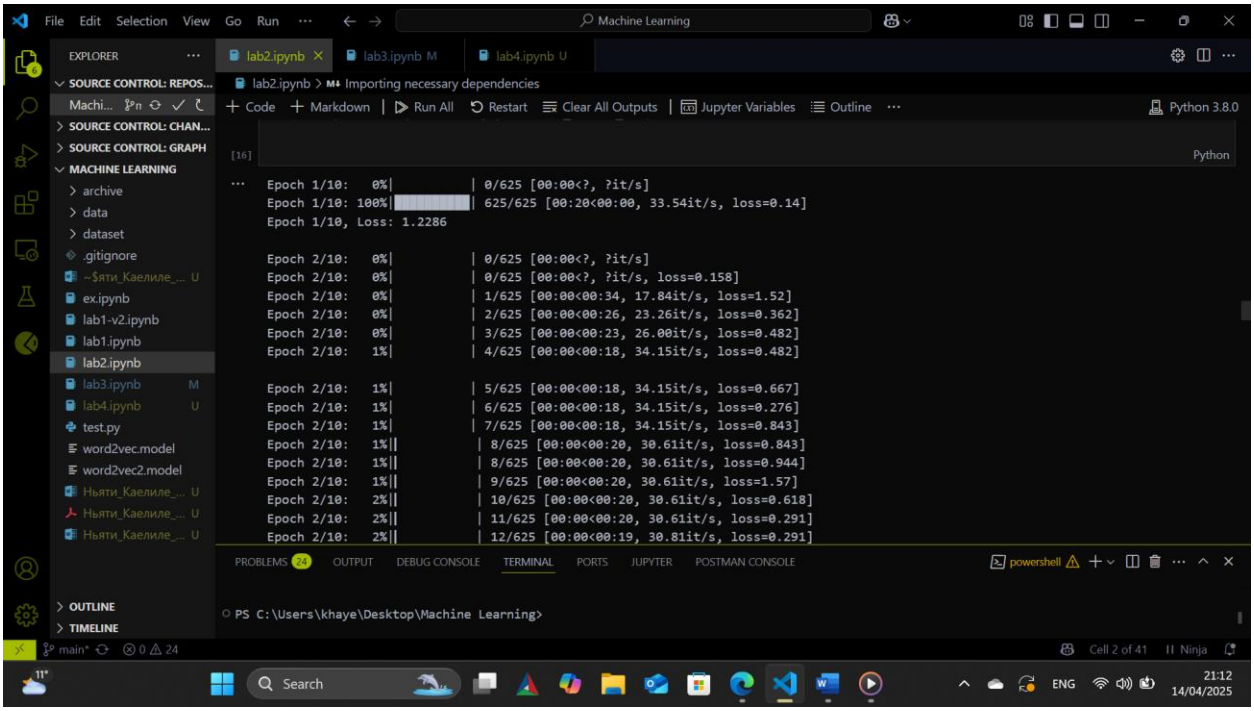


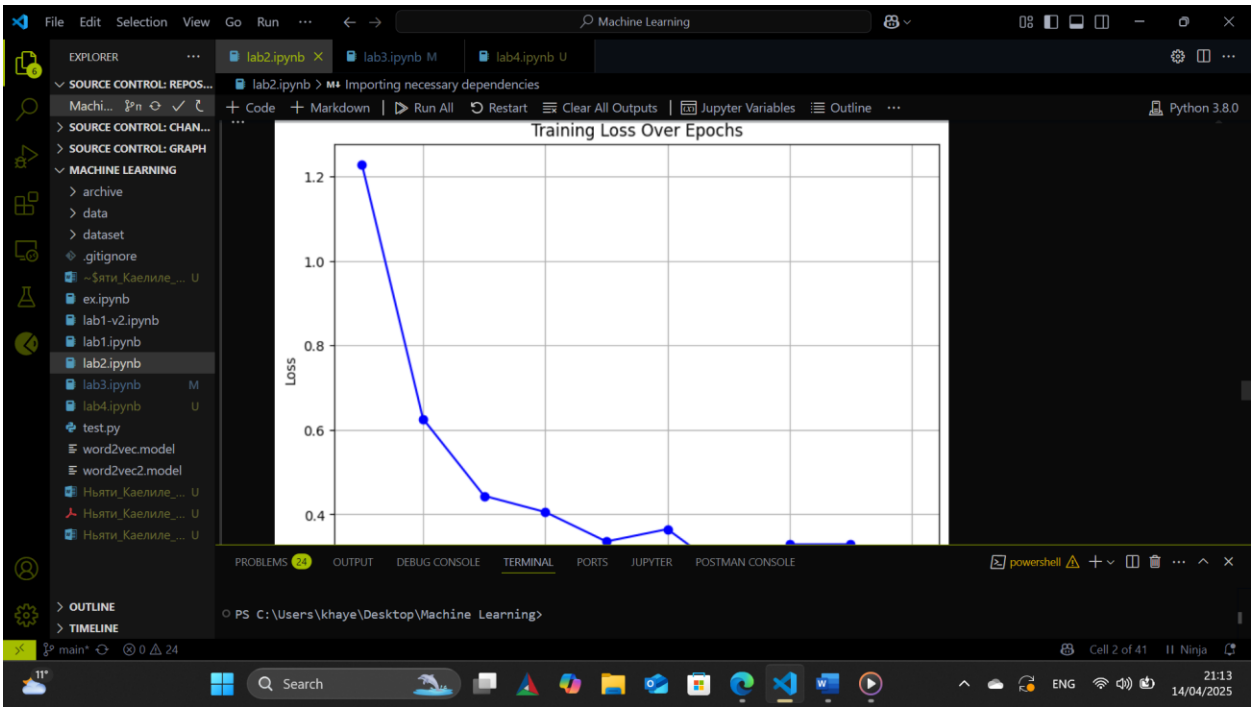
Рис 7. One Hot Encoding, ArgMax

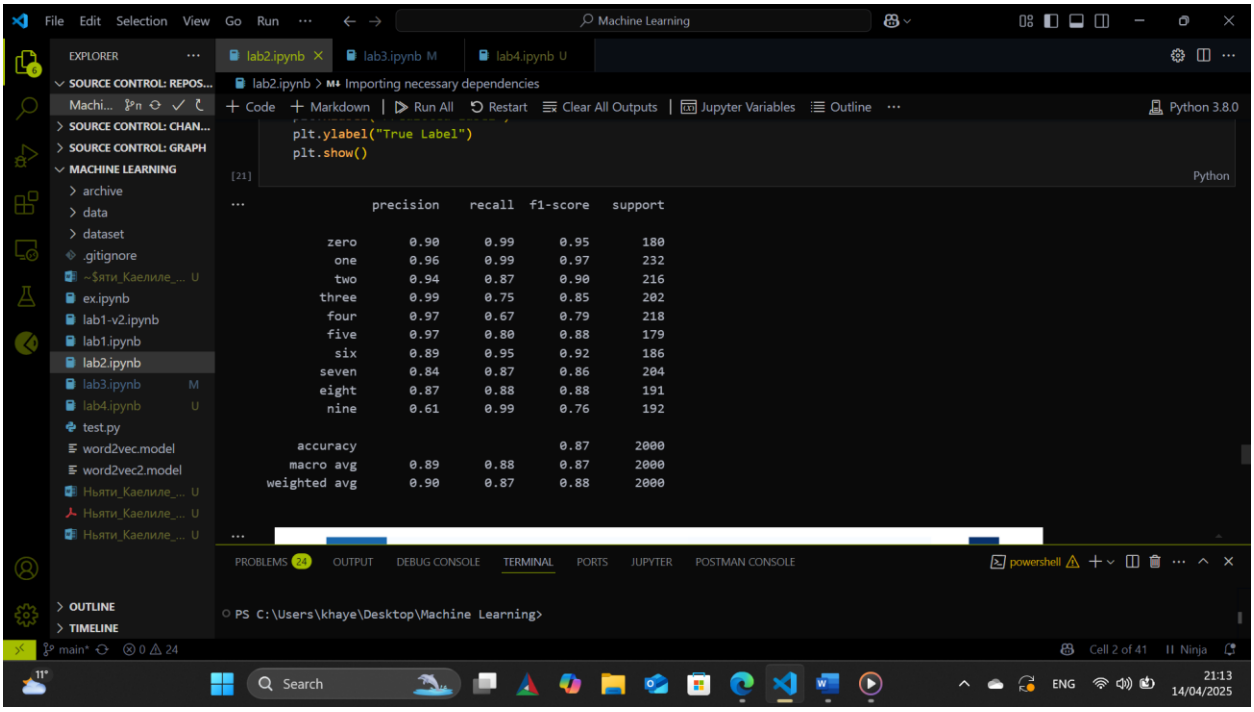

num_epochs = 10
loss_history = [] # Store loss for visualization
true_labels_OHE = []
predicted_labels_OHE = []
for epoch in range(num_epochs):
running_loss = 0.0
progress_bar = tqdm(train_loader, desc=f"Epoch {epoch+1}/{num_epochs}", leave=False) # Progress bar
for inputs, labels in train_loader:
inputs, labels = inputs.to(device), labels.to(device)
# One-hot encode the labels
one_hot_labels = one_hot_encode(labels).to(device)
optimizer.zero_grad()
outputs = model(inputs)
PS C:\Users\khaye\Desktop\Machine Learning>


#softmax = nn.Softmax(dim=1)
#probabilities = softmax(outputs)
Get predicted class indices
_, preds = torch.max(outputs, 1)
# Convert predictions to one-hot (optional, for comparison)
one_hot_preds = one_hot_encode(preds)
# Reverse transformation from one-hot to class index
predicted_class_indices = reverse_transform_one_hot(one_hot_preds)
# Store true and predicted labels
true_labels_OHE.extend(labels.cpu().numpy()) # Convert tensor to list
predicted_labels_OHE.extend(predicted_class_indices.cpu().numpy()) # Convert tensor to list
optimizer.step()
running_loss += loss.item() * inputs.size(0)
progress_bar.set_postfix(loss=loss.item()) # Show current loss
PS C:\Users\khaye\Desktop\Machine Learning>


progress_bar.set_postfix(loss=loss.item()) # Show current Loss
progress_bar.update(1)
epoch_loss = running_loss / len(train_subset)
loss_history.append(epoch_loss) # Store loss per epoch
print(f"Epoch {epoch+1}/{num_epochs}, Loss: {epoch_loss:.4f}")
# Plot Training Loss Curve
plt.figure(figsize=(8, 6))
plt.plot(range(1, num_epochs + 1), loss_history, marker='o', linestyle='-', color='b')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training Loss Over Epochs")
plt.grid()
plt.show()
print("One Hot true labels", true_labels_OHE)
print("One Hot predicted", predicted_labels_OHE)
Epoch 1/10: 0%| | 0/625 [00:00<?, ?it/s]
PS C:\Users\khaye\Desktop\Machine Learning>


Epoch 1/10: 0%| | 0/625 [00:00<?, ?it/s]
Epoch 1/10: 100%| | 625/625 [00:20<00:00, 33.54it/s, loss=0.14]
Epoch 1/10, Loss: 1.2286
Epoch 2/10: 0%| | 0/625 [00:00<?, ?it/s]
Epoch 2/10: 0%| | 0/625 [00:00<?, ?it/s, loss=0.158]
Epoch 2/10: 0%| | 1/625 [00:00<00:34, 17.84it/s, loss=1.52]
Epoch 2/10: 0%| | 2/625 [00:00<00:26, 23.26it/s, loss=0.362]
Epoch 2/10: 0%| | 3/625 [00:00<00:23, 26.00it/s, loss=0.482]
Epoch 2/10: 1%| | 4/625 [00:00<00:18, 34.15it/s, loss=0.482]
Epoch 2/10: 1%| | 5/625 [00:00<00:18, 34.15it/s, loss=0.667]
Epoch 2/10: 1%| | 6/625 [00:00<00:18, 34.15it/s, loss=0.276]
Epoch 2/10: 1%| | 7/625 [00:00<00:18, 34.15it/s, loss=0.843]
Epoch 2/10: 1%|| | 8/625 [00:00<00:20, 30.61it/s, loss=0.843]
Epoch 2/10: 1%|| | 8/625 [00:00<00:20, 30.61it/s, loss=0.944]
Epoch 2/10: 1%|| | 9/625 [00:00<00:20, 30.61it/s, loss=1.57]
Epoch 2/10: 2%|| | 10/625 [00:00<00:20, 30.61it/s, loss=0.618]
Epoch 2/10: 2%|| | 11/625 [00:00<00:20, 30.61it/s, loss=0.291]
Epoch 2/10: 2%|| | 12/625 [00:00<00:19, 30.81it/s, loss=0.291]
PS C:\Users\khaye\Desktop\Machine Learning>


Training Loss Over Epochs
Loss
1.2
1.0
0.8
0.6
0.4
Machine Learning
lab2.ipynb
lab3.ipynb M
lab4.ipynb U
Importing necessary dependencies
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3.8.0
PROBLEMS 24
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
JUPYTER
POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
Search
ENG
21:13
14/04/2025


plt.ylabel("True Label")
plt.show()
[21]
Python
precision recall f1-score support
zero 0.90 0.99 0.95 180
one 0.96 0.99 0.97 232
two 0.94 0.87 0.90 216
three 0.99 0.75 0.85 202
four 0.97 0.67 0.79 218
five 0.97 0.80 0.88 179
six 0.89 0.95 0.92 186
seven 0.84 0.87 0.86 204
eight 0.87 0.88 0.88 191
nine 0.61 0.99 0.76 192
accuracy 0.87 2000
macro avg 0.89 0.88 0.87 2000
weighted avg 0.90 0.87 0.88 2000
Machine Learning
lab2.ipynb
lab3.ipynb M
lab4.ipynb U
Importing necessary dependencies
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3.8.0
PROBLEMS 24
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
JUPYTER
POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
Search
ENG
21:13
14/04/2025

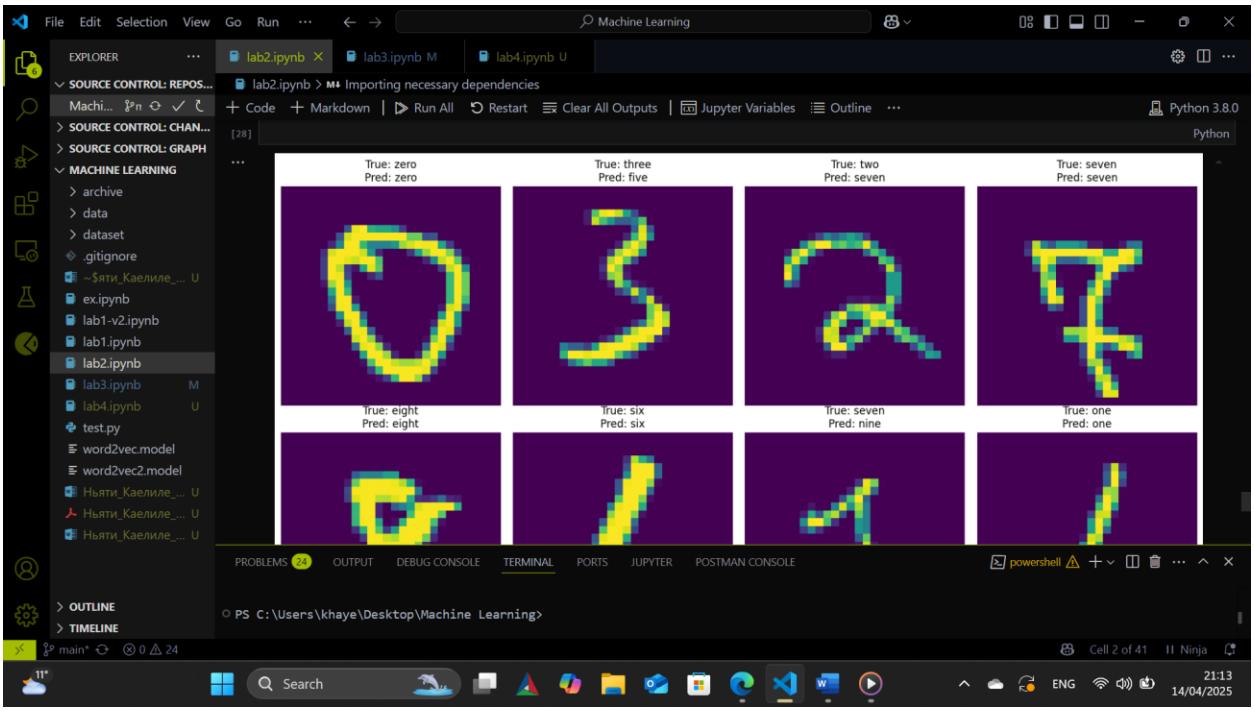


Рис 8. Результаты

3. Выводы

В ходе выполнения задания был использован датасет MNIST, содержащий изображения рукописных цифр, для обучения и тестирования алгоритмов классификации. Реализованный подход продемонстрировал способность эффективно распознавать цифры благодаря хорошо структурированным данным и применению современных методов машинного обучения. Результаты подтверждают пригодность выбранных алгоритмов для решения задач распознавания образов и подчеркивают важность качественных датасетов в процессе обучения моделей.