Задание.

1. Пополнить MNIST 11-м классом "не цифры", поместив в него случайно выбранные буквы EMNIST.

11-й класс формируется из представителей всех классов EMNIST.

В обучающую порцию данных 11-го класса добавляются по 231 примеру из первых 20-и классов EMNIST

и по 230 из последующих (231 * 20 + 230 * 6 = 6000).

Аналогично создается и проверочная порция данных 11-го класса MNIST.

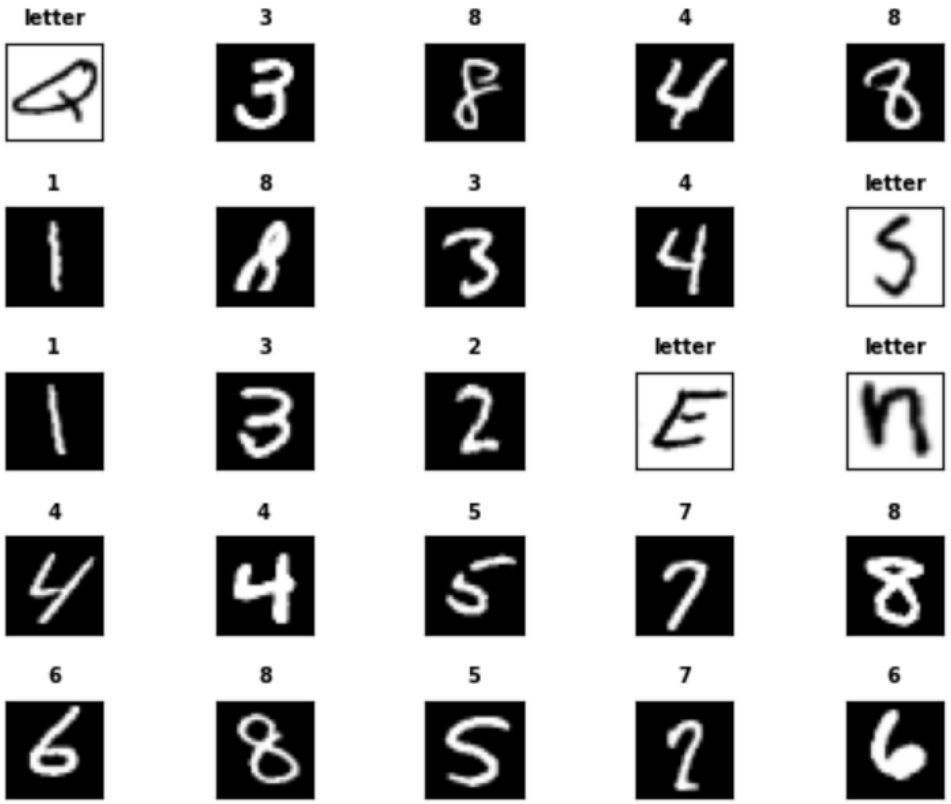
```
def add_20_class_with_231_elem(e_x_train, e_y_train, e_number_train):
    x = np.zeros(4620 * 28 * 28 * 1)
    x = x.reshape(4620, 28, 28, 1)
    for j in range(20):
        k = 0
        class_i = j
        fl = True
        i = 0
        while fl and (i < e_number_train):
            if (k == 231):
                fl = False
            else :
                if ((e_y_train[i]) == class_i):
                    x[k + 231 * j, ...] = e_x_train[i, ...]
                    k += 1
                i += 1
    return x

def add_6_class_with_230_elem(e_x_train, e_y_train, e_number_train):
    x = np.zeros(1380 * 28 * 28 * 1)
    x = x.reshape(1380, 28, 28, 1)
    for j in range(6):
        k = 0
        class_i =20 + j
        fl = True
        i = 20
        while fl and (i < e_number_train):
            if (k == 230):
                fl = False
            else :
                if ((e_y_train[i]) == class_i):
                    x[k + 230 * j, ...] = e_x_train[i, ...]
                    k += 1
                i += 1
    return x
def add_test(e_x_test, e_y_test, e_number_test):
    x = np.zeros(1300 * 28 * 28 * 1)
```

```
    x = x.reshape(1300, 28, 28, 1)
    for j in range(26):
        k = 0
        class_i = j
        fl = True
        i = 0
        while fl and (i < e_number_test):
            if (k == 50):
                fl = False
            else :
                if ((e_y_test[i]) == class_i):
                    x[k + 50*j, ...] = e_x_test[i, ...]
                    k += 1
                i += 1
    return x
x_20 = add_20_class_with_231_elem(e_x_train, e_y_train, e_number_train)
x_6 = add_6_class_with_230_elem(e_x_train, e_y_train, e_number_train)
x_11_class = np.vstack((x_20, x_6))
x_train = np.vstack((x_train, x_11_class))

#класс emnist-letters будет 10 классом, чтобы различаться от 0-9 классов цифр
y_11_class = np.ones(6000, dtype=np.int64)
y_11_class *= 10
y_train = np.concatenate((y_train, y_11_class))

#для тестового множества
x_11_test = add_test(e_x_test, e_y_test, e_number_test)
x_test = np.vstack((x_test, x_11_test))
y_11_test = np.ones(1300, dtype=np.int64)
y_11_test *= 10
y_test = np.concatenate((y_test, y_11_test))
OutputData25(x_test, y_test)
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```



2. Используя PyTorch, создать и обучить модель нейронной сети для классификации примеров сформированного в п. 1 набора данных.

Максимально приблизить точность классификации к 99,6%.

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=(5, 5))
        self.conv2 = nn.Conv2d(32, 64, kernel_size=(5, 5))
        self.max_pool2d1 = nn.MaxPool2d(2)
        self.max_pool2d2 = nn.MaxPool2d(2)
        self.conv2_drop = nn.Dropout2d(p=0.3)
        self.dropout = nn.Dropout(p=0.3)
        self.fc1 = nn.Linear(in_features=1024, out_features=16, bias=True)
        self.fc2 = nn.Linear(in_features=16, out_features=11, bias=True)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.max_pool2d1(x)
        x = self.conv2_drop(x)
        x = self.conv2(x)
        x = F.relu(x)
```

```
        x = self.max_pool2d2(x)  # torch.Size([256, 32, 5, 5])
        x = x.view(-1, 1024)
        x = self.dropout(x)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=-1)
```

```
Net(
  (conv1): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
  (max_pool2d1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (max_pool2d2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv2_drop): Dropout2d(p=0.3, inplace=False)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc1): Linear(in_features=1024, out_features=16, bias=True)
  (fc2): Linear(in_features=16, out_features=11, bias=True)
)
```

3. Обученную модель сохранить в файл.

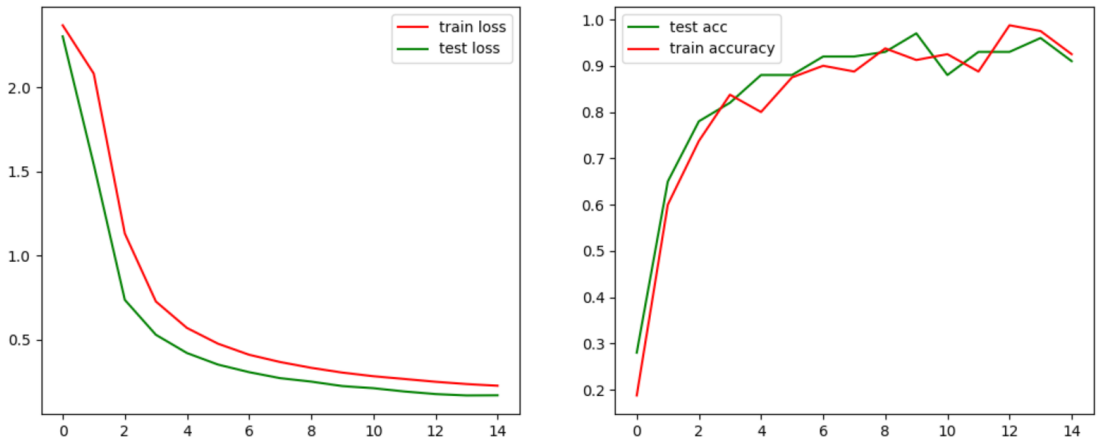
```
torch.save(model.state_dict(), 'model_weights.pth')
```

4. При обучении, кроме потерь, выводить точность.

Epoch: 20
Loss: train: 0.250135 test: 0.177098
Accuracy: train: 0.987500 test : 0.930000
Learning time: 36.608376264572144

---

5. Построить графики обучения (по аналогии с ЛР6).



6. В режиме проверки (прогнозирования) загружать модель из файла.

```
with torch.no_grad():
  model.load_state_dict(torch.load('model_weights.pth'))
  model.eval()
  output = model(torch.from_numpy(x_test.reshape(-1, 1, 28,
28).astype(np.float32)))
  output = output.detach().numpy()
  output = np.array([np.argmax(item) for item in output])
```

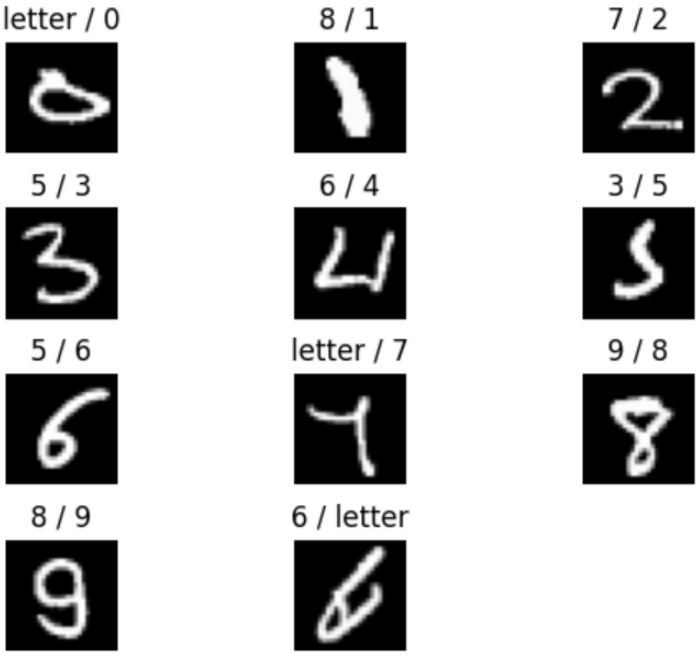
7. Вывести точность классификации примеров проверочного множества по классам.

Number 0 accuracy: 0.9785714285714285
Number 1 accuracy: 0.9876651982378855
Number 2 accuracy: 0.9486434108527132
Number 3 accuracy: 0.9554455445544554
Number 4 accuracy: 0.9439918533604889
Number 5 accuracy: 0.9517937219730942
Number 6 accuracy: 0.9707724425887265
Number 7 accuracy: 0.9319066147859922
Number 8 accuracy: 0.9291581108829569
Number 9 accuracy: 0.9444995044598612
Letter accuracy: 0.8876923076923077

8. Вывести оценки качества модели посредством classification_report
metrics.classification_report

	precision	recall	f1-score	support
0	0.94	0.98	0.96	980
1	0.98	0.99	0.98	1135
2	0.93	0.95	0.94	1032
3	0.94	0.96	0.95	1010
4	0.96	0.94	0.95	982
5	0.95	0.95	0.95	892
6	0.95	0.97	0.96	958
7	0.96	0.93	0.94	1028
8	0.94	0.93	0.94	974
9	0.93	0.94	0.94	1009
10	0.94	0.89	0.91	1300
accuracy			0.95	11300
macro avg	0.95	0.95	0.95	11300
weighted avg	0.95	0.95	0.95	11300

9. Вывести рисунки неверно классифицированных изображений (примеры).



10. Напечатать список неверно классифицированных изображений проверочного множества (в виде таблицы),
указывая в нем номер класса и индекс изображения. Список упорядочить по числу ошибок в классе.
number: 1, wrong index: 619, wrong Size: 14
number: 0, wrong index: 717, wrong Size: 21
number: 6, wrong index: 217, wrong Size: 28
number: 5, wrong index: 340, wrong Size: 43
number: 3, wrong index: 313, wrong Size: 45
number: 2, wrong index: 77, wrong Size: 53
number: 4, wrong index: 33, wrong Size: 55
number: 9, wrong index: 241, wrong Size: 56
number: 8, wrong index: 233, wrong Size: 69
number: 7, wrong index: 358, wrong Size: 70
number: 10, wrong index: 10010, wrong Size: 146