Genkendelse af håndskrift

In [1]:

```
!python.exe -m pip install --upgrade pip
!pip install torch
!pip install torchvision
!pip install matplotlib
```

```
Requirement already satisfied: pip in c:\users\chr_v\documents\eaa23itek\3semeste
r\kunstig-intelligens\ai\lib\site-packages (24.2)
Requirement already satisfied: torch in c:\users\chr_v\documents\eaa23itek\3semes
ter\kunstig-intelligens\ai\lib\site-packages (2.4.1)
Requirement already satisfied: filelock in c:\users\chr_v\documents\eaa23itek\3se
mester\kunstig-intelligens\ai\lib\site-packages (from torch) (3.16.0)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\chr_v\documen
ts\eaa23itek\3semester\kunstig-intelligens\ai\lib\site-packages (from torch) (4.1
2.2)
Requirement already satisfied: sympy in c:\users\chr_v\documents\eaa23itek\3semes
ter\kunstig-intelligens\ai\lib\site-packages (from torch) (1.13.2)
Requirement already satisfied: networkx in c:\users\chr_v\documents\eaa23itek\3se
mester\kunstig-intelligens\ai\lib\site-packages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in c:\users\chr_v\documents\eaa23itek\3seme
ster\kunstig-intelligens\ai\lib\site-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in c:\users\chr_v\documents\eaa23itek\3seme
ster\kunstig-intelligens\ai\lib\site-packages (from torch) (2024.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\chr_v\documents\eaa23i
tek\3semester\kunstig-intelligens\ai\lib\site-packages (from jinja2->torch) (2.1.
5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\chr_v\documents\eaa
23itek\3semester\kunstig-intelligens\ai\lib\site-packages (from sympy->torch) (1.
3.0)
Requirement already satisfied: torchvision in c:\users\chr_v\documents\eaa23itek
\3semester\kunstig-intelligens\ai\lib\site-packages (0.19.1)
Requirement already satisfied: numpy in c:\users\chr_v\documents\eaa23itek\3semes
ter\kunstig-intelligens\ai\lib\site-packages (from torchvision) (2.0.2)
Requirement already satisfied: torch==2.4.1 in c:\users\chr_v\documents\eaa23itek
\3semester\kunstig-intelligens\ai\lib\site-packages (from torchvision) (2.4.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in c:\users\chr_v\documents
\eaa23itek\3semester\kunstig-intelligens\ai\lib\site-packages (from torchvision)
(10.4.0)
Requirement already satisfied: filelock in c:\users\chr_v\documents\eaa23itek\3se
mester\kunstig-intelligens\ai\lib\site-packages (from torch==2.4.1->torchvision)
(3.16.0)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\chr_v\documen
ts\eaa23itek\3semester\kunstig-intelligens\ai\lib\site-packages (from torch==2.4.
1->torchvision) (4.12.2)
Requirement already satisfied: sympy in c:\users\chr_v\documents\eaa23itek\3semes
ter\kunstig-intelligens\ai\lib\site-packages (from torch==2.4.1->torchvision) (1.
13.2)
Requirement already satisfied: networkx in c:\users\chr_v\documents\eaa23itek\3se
mester\kunstig-intelligens\ai\lib\site-packages (from torch==2.4.1->torchvision)
(3.2.1)
Requirement already satisfied: jinja2 in c:\users\chr_v\documents\eaa23itek\3seme
ster\kunstig-intelligens\ai\lib\site-packages (from torch==2.4.1->torchvision)
(3.1.4)
Requirement already satisfied: fsspec in c:\users\chr_v\documents\eaa23itek\3seme
ster\kunstig-intelligens\ai\lib\site-packages (from torch==2.4.1->torchvision) (2
024.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\chr_v\documents\eaa23i
tek\3semester\kunstig-intelligens\ai\lib\site-packages (from jinja2->torch==2.4.1
->torchvision) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\chr_v\documents\eaa
23itek\3semester\kunstig-intelligens\ai\lib\site-packages (from sympy->torch==2.
4.1->torchvision) (1.3.0)
Requirement already satisfied: matplotlib in c:\users\chr_v\documents\eaa23itek\3
semester\kunstig-intelligens\ai\lib\site-packages (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\chr_v\documents\eaa23
itek\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (1.3.0)
```

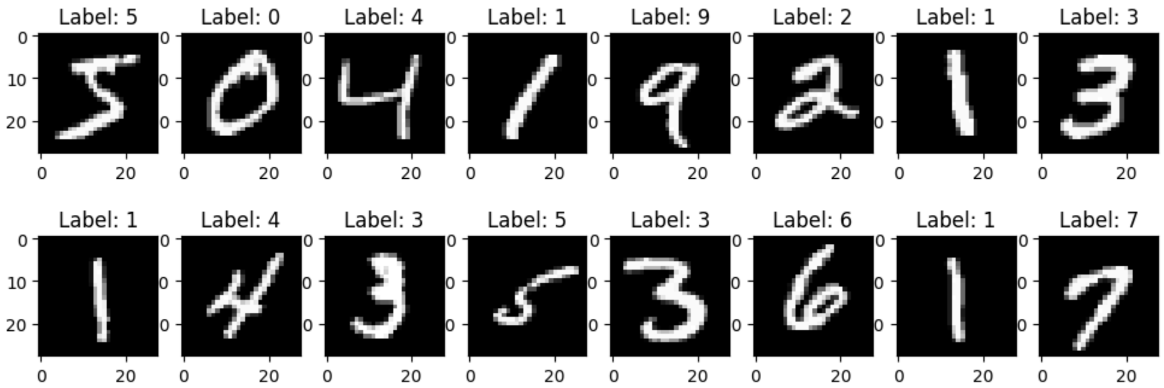
```
Requirement already satisfied: cycler>=0.10 in c:\users\chr_v\documents\eaa23itek
\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\chr_v\documents\eaa2
3itek\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (4.53.
1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\chr_v\documents\eaa2
3itek\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (1.4.
5)
Requirement already satisfied: numpy>=1.23 in c:\users\chr_v\documents\eaa23itek
\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in c:\users\chr_v\documents\eaa23i
tek\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\chr_v\documents\eaa23itek\3s
emester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\chr_v\documents\eaa23
itek\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\chr_v\documents\e
aa23itek\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotlib) (2.
9.0.post0)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\chr_v\docum
ents\eaa23itek\3semester\kunstig-intelligens\ai\lib\site-packages (from matplotli
b) (6.4.4)
Requirement already satisfied: zipp>=3.1.0 in c:\users\chr_v\documents\eaa23itek
\3semester\kunstig-intelligens\ai\lib\site-packages (from importlib-resources>=3.
2.0->matplotlib) (3.20.1)
Requirement already satisfied: six>=1.5 in c:\users\chr_v\documents\eaa23itek\3se
mester\kunstig-intelligens\ai\lib\site-packages (from python-dateutil>=2.7->matpl
otlib) (1.16.0)
```

In [2]:

```python
import torch
from torchvision import datasets, transforms
import matplotlib.pyplot as plt

# Transform PIL image into a tensor. The values are in the range [0, 1]
t = transforms.ToTensor()

# Load datasets for training and testing.
mnist_training = datasets.MNIST(root='/tmp/mnist', train=True, download=True, tr
mnist_val = datasets.MNIST(root='/tmp/mnist', train=False, download=True, transf
```

In [3]:

```python
# Plot some digits.
cols = 8
rows = 2

fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(1.5*cols, 2*rows))
for i, ax in enumerate(axes.flatten()):
    image, label = mnist_training[i] # returns PIL image with its labels
    ax.set_title(f"Label: {label}")
    ax.imshow(image.squeeze(0), cmap='gray') # we get a 1x28x28 tensor -> remove
plt.show()
```



Der er hermed importeret de relevante libraries og loaded den ønskede data. Der er både loaded træningsdata og testdata. Ovenfor ses hvordan forskellige tal kan vises som en 28x28 pixels.

In [4]:

```
mnist_training[317]
# Label er 2
print('Label til nummer 317: ', mnist_training[317][1])

# antal sorte pixellinjer
number_p = 0
for x in mnist_training[317][0][0]:
    #print(x)
    if sum(x) == 0: # hvidt -> 1
        number_p += 1

print('Antal pixellinjer, der er sorte: ', number_p) # der er 20 pixellinjer, de
```

```
Label til nummer 317:  2
Antal pixellinjer, der er sorte:  8
```

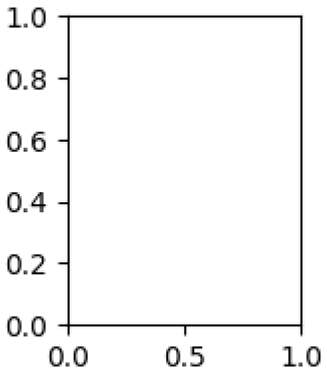
Heraf ses at der er 20 sorte pixellinjer.

Label til sampel 317 er 2.

Nedenfor er sampel 317 plottet.

In [5]:

```
fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(1.5, 2))
image, label = mnist_training[317] # returns PIL image with its labels

ax = axes.flatten()
ax.set_title(f"Label: {label}")
ax.imshow(image.squeeze(0), cmap='gray') # we get a 1x28x28 tensor -> remove fir
plt.show()
```



In [6]:

```
def model_A():
    model = torch.nn.Sequential(
        torch.nn.Linear(28*28, 256),
        torch.nn.ReLU(),
        torch.nn.Linear(256, 10)
    )

    # Use Adam as optimizer.
    opt = torch.optim.Adam(params=model.parameters(), lr=0.01)

    # Use CrossEntropyLoss for as loss function.
    loss_fn = torch.nn.CrossEntropyLoss()
    return model, opt, loss_fn
```

In [7]:

```
# We train the model with batches of 500 examples.

def train_model(data, model, batch_size, epochs):
    model, opt, loss_fn = model
    train_loader = torch.utils.data.DataLoader(data, batch_size=batch_size, shuf

    losses = []

    for epoch in range(epochs):
        for imgs, labels in train_loader:
            n = len(imgs)
            # Reshape data from [500, 1, 28, 28] to [500, 784] and use the model
            predictions = model(imgs.view(n, -1))
            # Compute the loss.
            loss = loss_fn(predictions, labels)
            opt.zero_grad()
            loss.backward()
            opt.step()
            losses.append(float(loss))
        print(f"Epoch: {epoch}, Loss: {float(loss)}")

    return model

model_500_10 = train_model(mnist_training, model_A(), 500, 10)
```

```
Epoch: 0, Loss: 0.17375962436199188
Epoch: 1, Loss: 0.06520172953605652
Epoch: 2, Loss: 0.08400581032037735
Epoch: 3, Loss: 0.03866270184516907
Epoch: 4, Loss: 0.04371411353349686
Epoch: 5, Loss: 0.07129082083702087
Epoch: 6, Loss: 0.032782480120658875
Epoch: 7, Loss: 0.029134400188922882
Epoch: 8, Loss: 0.015331805683672428
Epoch: 9, Loss: 0.037380997091531754
```

In [8]:

```
model_500_30 = train_model(mnist_training, model_A(), 500, 30)
```

```
Epoch: 0, Loss: 0.12886908650398254
Epoch: 1, Loss: 0.0940222516655922
Epoch: 2, Loss: 0.054536186158657074
Epoch: 3, Loss: 0.04875937104225159
Epoch: 4, Loss: 0.053535547107458115
Epoch: 5, Loss: 0.02680111862719059
Epoch: 6, Loss: 0.040136463940143585
Epoch: 7, Loss: 0.03649178147315979
Epoch: 8, Loss: 0.0276471134275198
Epoch: 9, Loss: 0.0675947517156601
Epoch: 10, Loss: 0.012285485863685608
Epoch: 11, Loss: 0.01712060160934925
Epoch: 12, Loss: 0.005569661967456341
Epoch: 13, Loss: 0.007899112068116665
Epoch: 14, Loss: 0.009004170075058937
Epoch: 15, Loss: 0.020622342824935913
Epoch: 16, Loss: 0.0197784174233675
Epoch: 17, Loss: 0.004378194455057383
Epoch: 18, Loss: 0.009863585233688354
Epoch: 19, Loss: 0.033213965594768524
Epoch: 20, Loss: 0.018535476177930832
Epoch: 21, Loss: 0.0034608745481818914
Epoch: 22, Loss: 0.015533316880464554
Epoch: 23, Loss: 0.022409431636333466
Epoch: 24, Loss: 0.020983422175049782
Epoch: 25, Loss: 0.001446520909667015
Epoch: 26, Loss: 0.019843781366944313
Epoch: 27, Loss: 0.0033087972551584244
Epoch: 28, Loss: 0.018927963450551033
Epoch: 29, Loss: 5.011044049751945e-05
```

In [9]:
```
model_1500_10 = train_model(mnist_training, model_A(), 1500, 10)
```

```
Epoch: 0, Loss: 0.22000446915626526
Epoch: 1, Loss: 0.11750645190477371
Epoch: 2, Loss: 0.09253121167421341
Epoch: 3, Loss: 0.0503368116915226
Epoch: 4, Loss: 0.05059480294585228
Epoch: 5, Loss: 0.041387323290109634
Epoch: 6, Loss: 0.03682173788547516
Epoch: 7, Loss: 0.024217844009399414
Epoch: 8, Loss: 0.02228965237736702
Epoch: 9, Loss: 0.016132861375808716
```

In [10]:
```
model_1500_20 = train_model(mnist_training, model_A(), 1500, 20)
```

```
Epoch: 0, Loss: 0.197243332862854
Epoch: 1, Loss: 0.12457828968763351
Epoch: 2, Loss: 0.08496056497097015
Epoch: 3, Loss: 0.08109144866466522
Epoch: 4, Loss: 0.043951649218797684
Epoch: 5, Loss: 0.033044878393411636
Epoch: 6, Loss: 0.03875336796045303
Epoch: 7, Loss: 0.027706729248166084
Epoch: 8, Loss: 0.01854648068547249
Epoch: 9, Loss: 0.01453008409589529
Epoch: 10, Loss: 0.01667616330087185
Epoch: 11, Loss: 0.0064813969656825066
Epoch: 12, Loss: 0.00575874000787735
Epoch: 13, Loss: 0.004075347445905209
Epoch: 14, Loss: 0.003458729712292552
Epoch: 15, Loss: 0.0020745352376252413
Epoch: 16, Loss: 0.0019065478118136525
Epoch: 17, Loss: 0.001259386190213263
Epoch: 18, Loss: 0.0014032196486368775
Epoch: 19, Loss: 0.0008221659809350967
```

In [11]:

```
# Determine the accuracy of our classifier
# Load all 10000 images from the validation set.
n = 10000
loader = torch.utils.data.DataLoader(mnist_val, batch_size=n)

#images, labels = iter(loader).next()
images, labels = next(iter(loader))

# The tensor images has the shape [10000, 1, 28, 28]. Reshape the tensor to
# [10000, 784] as our model expected a flat vector.
data = images.view(n, -1)
```

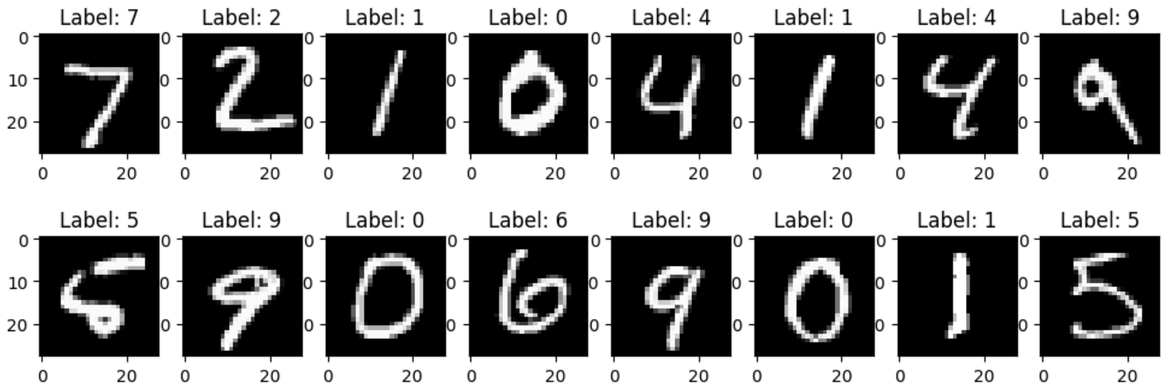
In [12]:

```
def test_model(data, labels, model):
    predictions = model(data)
    predicted_classes = torch.argmax(predictions, dim=1)

    accuracy = sum(predicted_classes.numpy() == labels.numpy()) / 10000
    print(accuracy)
    return predicted_classes, accuracy

test_model(data, labels, model_500_10)
test_model(data, labels, model_500_30)
test_model(data, labels, model_1500_10)
predicted_classes, ac = test_model(data, labels, model_1500_20)
```

```
0.9752
0.979
0.9788
0.9801
```

In [13]:

```
cols = 8
rows = 2

fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(1.5*cols, 2*rows))
for i, ax in enumerate(axes.flatten()):
    image, label = images[i],labels[i]          # returns PIL image with its lab
    ax.set_title(f"Label: {label}")
    ax.imshow(image.squeeze(0), cmap='gray')  # we get a 1x28x28 tensor -> remov
plt.show()
```



In [14]:

```
errors = []

print(" i, prediction, label")
for n in range(10000):
    if (predicted_classes[n] != labels[n]):
        print(n,predicted_classes[n],labels[n])
        errors.append(n)

print("length:",errors.__len__())
```

```
 i, prediction, label
149 tensor(9) tensor(2)
247 tensor(2) tensor(4)
321 tensor(7) tensor(2)
340 tensor(3) tensor(5)
445 tensor(0) tensor(6)
495 tensor(2) tensor(8)
551 tensor(3) tensor(7)
582 tensor(2) tensor(8)
619 tensor(8) tensor(1)
684 tensor(3) tensor(7)
691 tensor(4) tensor(8)
720 tensor(8) tensor(5)
882 tensor(7) tensor(9)
900 tensor(3) tensor(1)
947 tensor(9) tensor(8)
951 tensor(4) tensor(5)
956 tensor(2) tensor(1)
965 tensor(0) tensor(6)
1014 tensor(5) tensor(6)
1039 tensor(2) tensor(7)
1044 tensor(8) tensor(6)
1112 tensor(6) tensor(4)
1156 tensor(8) tensor(7)
1182 tensor(5) tensor(6)
1194 tensor(9) tensor(7)
1226 tensor(2) tensor(7)
1232 tensor(4) tensor(9)
1242 tensor(9) tensor(4)
1247 tensor(5) tensor(9)
1319 tensor(3) tensor(8)
1328 tensor(9) tensor(7)
1393 tensor(3) tensor(5)
1395 tensor(3) tensor(2)
1500 tensor(4) tensor(7)
1522 tensor(9) tensor(7)
1530 tensor(7) tensor(8)
1549 tensor(6) tensor(4)
1553 tensor(3) tensor(9)
1609 tensor(6) tensor(2)
1642 tensor(6) tensor(2)
1670 tensor(3) tensor(5)
1678 tensor(0) tensor(2)
1681 tensor(7) tensor(3)
1717 tensor(0) tensor(8)
1751 tensor(2) tensor(4)
1754 tensor(2) tensor(7)
1790 tensor(8) tensor(2)
1901 tensor(4) tensor(9)
1913 tensor(8) tensor(3)
1982 tensor(5) tensor(6)
1984 tensor(0) tensor(2)
2004 tensor(3) tensor(8)
2024 tensor(9) tensor(7)
2035 tensor(3) tensor(5)
2053 tensor(9) tensor(4)
2070 tensor(9) tensor(7)
2098 tensor(0) tensor(2)
2109 tensor(9) tensor(3)
2118 tensor(0) tensor(6)
```

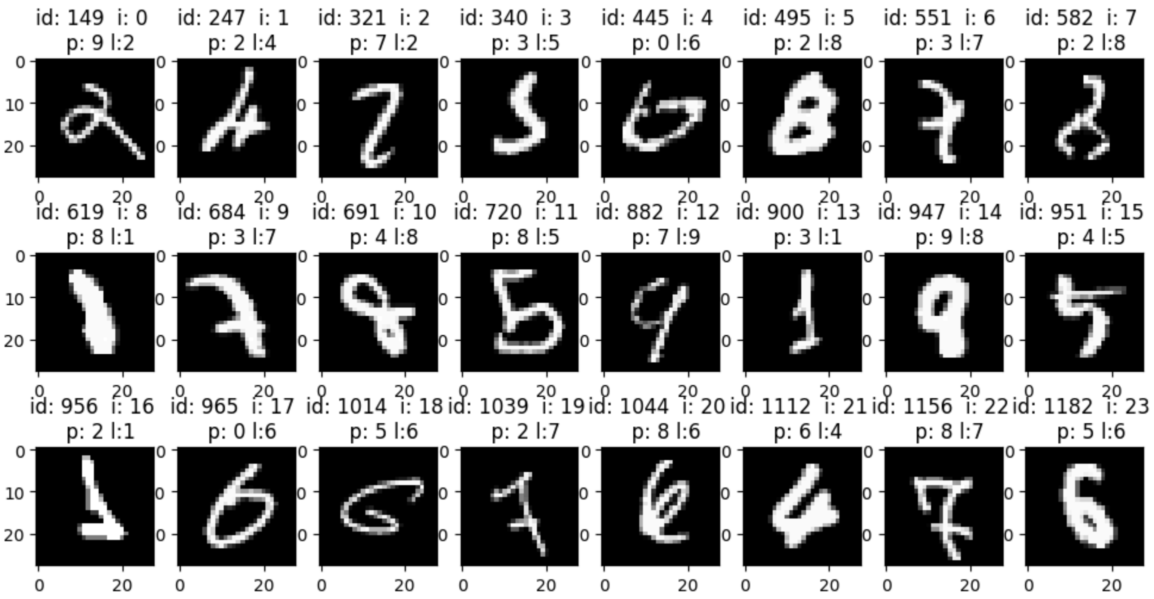
```
2130 tensor(9) tensor(4)
2135 tensor(1) tensor(6)
2182 tensor(2) tensor(1)
2272 tensor(0) tensor(8)
2293 tensor(6) tensor(9)
2369 tensor(7) tensor(5)
2387 tensor(1) tensor(9)
2406 tensor(8) tensor(9)
2488 tensor(4) tensor(2)
2526 tensor(3) tensor(5)
2597 tensor(3) tensor(5)
2607 tensor(1) tensor(7)
2648 tensor(0) tensor(9)
2654 tensor(1) tensor(6)
2720 tensor(4) tensor(9)
2810 tensor(3) tensor(5)
2836 tensor(7) tensor(4)
2863 tensor(4) tensor(9)
2877 tensor(9) tensor(4)
2896 tensor(0) tensor(8)
2915 tensor(3) tensor(7)
2921 tensor(2) tensor(3)
2939 tensor(7) tensor(9)
3060 tensor(7) tensor(9)
3073 tensor(2) tensor(1)
3117 tensor(9) tensor(5)
3289 tensor(9) tensor(8)
3405 tensor(9) tensor(4)
3422 tensor(0) tensor(6)
3503 tensor(1) tensor(9)
3520 tensor(4) tensor(6)
3558 tensor(0) tensor(5)
3567 tensor(5) tensor(8)
3597 tensor(3) tensor(9)
3681 tensor(3) tensor(2)
3718 tensor(9) tensor(4)
3727 tensor(4) tensor(8)
3749 tensor(4) tensor(6)
3751 tensor(2) tensor(7)
3757 tensor(3) tensor(8)
3776 tensor(8) tensor(5)
3818 tensor(4) tensor(0)
3853 tensor(2) tensor(6)
3893 tensor(6) tensor(5)
3906 tensor(3) tensor(1)
3943 tensor(5) tensor(3)
3985 tensor(4) tensor(9)
4065 tensor(8) tensor(0)
4078 tensor(3) tensor(9)
4156 tensor(3) tensor(2)
4199 tensor(9) tensor(7)
4201 tensor(7) tensor(1)
4248 tensor(8) tensor(2)
4271 tensor(3) tensor(5)
4289 tensor(8) tensor(2)
4294 tensor(5) tensor(9)
4425 tensor(4) tensor(9)
4437 tensor(2) tensor(3)
4443 tensor(2) tensor(3)
4497 tensor(7) tensor(8)
```

```
4500 tensor(1) tensor(9)
4534 tensor(8) tensor(9)
4536 tensor(5) tensor(6)
4548 tensor(6) tensor(5)
4551 tensor(4) tensor(7)
4571 tensor(8) tensor(6)
4740 tensor(5) tensor(3)
4761 tensor(8) tensor(9)
4807 tensor(3) tensor(8)
4814 tensor(4) tensor(6)
4823 tensor(4) tensor(9)
4860 tensor(9) tensor(4)
4876 tensor(4) tensor(2)
4880 tensor(8) tensor(0)
4956 tensor(4) tensor(8)
5078 tensor(8) tensor(3)
5331 tensor(6) tensor(1)
5457 tensor(8) tensor(1)
5634 tensor(3) tensor(2)
5642 tensor(5) tensor(1)
5676 tensor(3) tensor(4)
5749 tensor(5) tensor(8)
5888 tensor(0) tensor(4)
5936 tensor(9) tensor(4)
5937 tensor(3) tensor(5)
5955 tensor(9) tensor(3)
5972 tensor(3) tensor(5)
5973 tensor(9) tensor(3)
5982 tensor(3) tensor(5)
5997 tensor(9) tensor(5)
6009 tensor(9) tensor(3)
6011 tensor(9) tensor(3)
6023 tensor(9) tensor(3)
6059 tensor(9) tensor(3)
6166 tensor(3) tensor(9)
6555 tensor(9) tensor(8)
6571 tensor(7) tensor(9)
6574 tensor(6) tensor(2)
6597 tensor(7) tensor(0)
6625 tensor(4) tensor(8)
6651 tensor(5) tensor(0)
6755 tensor(7) tensor(8)
6783 tensor(6) tensor(1)
6847 tensor(4) tensor(6)
7216 tensor(6) tensor(0)
7434 tensor(8) tensor(4)
7451 tensor(6) tensor(5)
7800 tensor(2) tensor(3)
7823 tensor(2) tensor(8)
7921 tensor(2) tensor(8)
8062 tensor(8) tensor(5)
8091 tensor(8) tensor(2)
8094 tensor(8) tensor(2)
8255 tensor(0) tensor(4)
8311 tensor(4) tensor(6)
8325 tensor(6) tensor(0)
8522 tensor(6) tensor(8)
8527 tensor(9) tensor(4)
9009 tensor(2) tensor(7)
9015 tensor(2) tensor(7)
```

```
9024 tensor(2) tensor(7)
9280 tensor(5) tensor(8)
9500 tensor(7) tensor(2)
9587 tensor(4) tensor(9)
9634 tensor(8) tensor(0)
9664 tensor(7) tensor(2)
9669 tensor(7) tensor(4)
9679 tensor(3) tensor(6)
9692 tensor(7) tensor(9)
9700 tensor(8) tensor(2)
9729 tensor(6) tensor(5)
9745 tensor(0) tensor(4)
9749 tensor(6) tensor(5)
9770 tensor(0) tensor(5)
9779 tensor(0) tensor(2)
9792 tensor(7) tensor(4)
9793 tensor(5) tensor(6)
9839 tensor(3) tensor(2)
9941 tensor(8) tensor(5)
9944 tensor(8) tensor(3)
length:  199
```

In [15]:

```python
cols = 8
rows = 3

fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(1.5*cols, 2*rows))
for i, ax in enumerate(axes.flatten()):
    image, label = images[errors[i]],labels[predicted_classes[errors[i]]]
    ax.set_title(f"id: {errors[i]}  i: {i} \n  p: {predicted_classes[errors[i]]}
    ax.imshow(image.squeeze(0), cmap='gray')  # we get a 1x28x28 tensor -> remov
plt.show()
```



In [16]:

```python
def model_B():
    model = torch.nn.Sequential(
        torch.nn.Linear(28*28, 256),
        torch.nn.ReLU(),
        torch.nn.Linear(256, 256),
        torch.nn.ReLU(),
        torch.nn.Linear(256, 10)
    )
    # Use Adam as optimizer.
    opt = torch.optim.Adam(params=model.parameters(), lr=0.01)

    # Use CrossEntropyLoss for as loss function.
    loss_fn = torch.nn.CrossEntropyLoss()
    return model, opt, loss_fn

model_B_1500_60 = train_model(mnist_training, model_B(), 1500, 60)
```

```
Epoch: 0, Loss: 0.14487305283546448
Epoch: 1, Loss: 0.08681368827819824
Epoch: 2, Loss: 0.09352164715528488
Epoch: 3, Loss: 0.04889266565442085
Epoch: 4, Loss: 0.030047021806240082
Epoch: 5, Loss: 0.030225692316889763
Epoch: 6, Loss: 0.02593766525387764
Epoch: 7, Loss: 0.021595412865281105
Epoch: 8, Loss: 0.01988934352993965
Epoch: 9, Loss: 0.027679024264216423
Epoch: 10, Loss: 0.012619697488844395
Epoch: 11, Loss: 0.029425496235489845
Epoch: 12, Loss: 0.019025199115276337
Epoch: 13, Loss: 0.014553522691130638
Epoch: 14, Loss: 0.01590166799724102
Epoch: 15, Loss: 0.004444179590791464
Epoch: 16, Loss: 0.015438499860465527
Epoch: 17, Loss: 0.0016094620805233717
Epoch: 18, Loss: 0.007281886879354715
Epoch: 19, Loss: 0.009540324099361897
Epoch: 20, Loss: 0.017928017303347588
Epoch: 21, Loss: 0.014989101327955723
Epoch: 22, Loss: 0.019605042412877083
Epoch: 23, Loss: 0.008133887313306332
Epoch: 24, Loss: 0.011552699841558933
Epoch: 25, Loss: 0.013979545794427395
Epoch: 26, Loss: 0.014223241247236729
Epoch: 27, Loss: 0.007288345135748386
Epoch: 28, Loss: 0.0034425370395183563
Epoch: 29, Loss: 0.004972254857420921
Epoch: 30, Loss: 0.00850007776170969
Epoch: 31, Loss: 0.0047492943704128265
Epoch: 32, Loss: 0.010425703600049019
Epoch: 33, Loss: 0.008228140883147717
Epoch: 34, Loss: 0.01663164421916008
Epoch: 35, Loss: 0.0033757539931684732
Epoch: 36, Loss: 0.0023597110994160175
Epoch: 37, Loss: 0.002581906272098422
Epoch: 38, Loss: 0.006416057702153921
Epoch: 39, Loss: 0.011321030557155609
Epoch: 40, Loss: 0.004935507662594318
Epoch: 41, Loss: 0.012489015236496925
Epoch: 42, Loss: 0.012984877452254295
Epoch: 43, Loss: 0.015287664718925953
Epoch: 44, Loss: 0.01413290947675705
Epoch: 45, Loss: 0.035370517522096634
Epoch: 46, Loss: 0.01611599512398243
Epoch: 47, Loss: 0.018792882561683655
Epoch: 48, Loss: 0.0014396223705261946
Epoch: 49, Loss: 0.003648695070296526
Epoch: 50, Loss: 0.0064413282088935375
Epoch: 51, Loss: 0.01797708496451378
Epoch: 52, Loss: 0.005225990898907185
Epoch: 53, Loss: 0.00613600667566061
Epoch: 54, Loss: 0.009553739801049232
Epoch: 55, Loss: 0.015097490511834621
Epoch: 56, Loss: 0.021229494363069534
Epoch: 57, Loss: 0.012051062658429146
Epoch: 58, Loss: 0.006719444412738085
Epoch: 59, Loss: 0.018217874690890312
```

In [22]:

```
pred_classes, accuracy = test_model(data, labels, model_B_1500_60)
#print(pred_classes)

errors = []

print(" i, prediction, label")
for n in range(10000):
    if (pred_classes[n] != labels[n]):
        print(n, pred_classes[n], labels[n])
        errors.append(n)

print("length:",errors.__len__())
```

```
0.9723
tensor([7, 2, 1,  ..., 4, 5, 6])
 i, prediction, label
8 tensor(6) tensor(5)
104 tensor(5) tensor(9)
124 tensor(4) tensor(7)
149 tensor(9) tensor(2)
151 tensor(8) tensor(9)
247 tensor(2) tensor(4)
340 tensor(3) tensor(5)
359 tensor(8) tensor(9)
445 tensor(0) tensor(6)
448 tensor(8) tensor(9)
450 tensor(9) tensor(3)
582 tensor(2) tensor(8)
659 tensor(8) tensor(2)
717 tensor(6) tensor(0)
720 tensor(8) tensor(5)
726 tensor(9) tensor(7)
866 tensor(6) tensor(5)
874 tensor(4) tensor(9)
883 tensor(5) tensor(3)
951 tensor(4) tensor(5)
995 tensor(0) tensor(2)
1003 tensor(3) tensor(5)
1014 tensor(5) tensor(6)
1032 tensor(8) tensor(5)
1039 tensor(8) tensor(7)
1044 tensor(8) tensor(6)
1107 tensor(8) tensor(9)
1112 tensor(6) tensor(4)
1128 tensor(2) tensor(3)
1156 tensor(8) tensor(7)
1182 tensor(8) tensor(6)
1194 tensor(3) tensor(7)
1226 tensor(2) tensor(7)
1232 tensor(6) tensor(9)
1242 tensor(9) tensor(4)
1247 tensor(5) tensor(9)
1260 tensor(1) tensor(7)
1272 tensor(8) tensor(5)
1290 tensor(9) tensor(3)
1319 tensor(3) tensor(8)
1328 tensor(8) tensor(7)
1352 tensor(0) tensor(2)
1355 tensor(4) tensor(7)
1378 tensor(6) tensor(5)
1393 tensor(3) tensor(5)
1395 tensor(3) tensor(2)
1403 tensor(8) tensor(1)
1466 tensor(0) tensor(5)
1476 tensor(3) tensor(5)
1494 tensor(0) tensor(7)
1522 tensor(9) tensor(7)
1530 tensor(7) tensor(8)
1549 tensor(6) tensor(4)
1554 tensor(8) tensor(9)
1581 tensor(3) tensor(7)
1607 tensor(8) tensor(3)
1609 tensor(6) tensor(2)
```

```
1611 tensor(8) tensor(3)
1621 tensor(6) tensor(0)
1670 tensor(3) tensor(5)
1709 tensor(8) tensor(9)
1721 tensor(4) tensor(7)
1730 tensor(8) tensor(3)
1748 tensor(7) tensor(0)
1754 tensor(2) tensor(7)
1790 tensor(8) tensor(2)
1828 tensor(2) tensor(3)
1874 tensor(6) tensor(5)
1878 tensor(3) tensor(8)
1901 tensor(4) tensor(9)
1911 tensor(6) tensor(5)
1913 tensor(8) tensor(3)
1941 tensor(3) tensor(7)
1978 tensor(8) tensor(4)
2004 tensor(3) tensor(8)
2016 tensor(3) tensor(7)
2024 tensor(9) tensor(7)
2035 tensor(3) tensor(5)
2053 tensor(9) tensor(4)
2063 tensor(3) tensor(7)
2070 tensor(9) tensor(7)
2073 tensor(6) tensor(5)
2098 tensor(0) tensor(2)
2109 tensor(8) tensor(3)
2118 tensor(4) tensor(6)
2135 tensor(1) tensor(6)
2182 tensor(2) tensor(1)
2224 tensor(6) tensor(5)
2272 tensor(0) tensor(8)
2280 tensor(8) tensor(3)
2291 tensor(3) tensor(5)
2292 tensor(5) tensor(9)
2293 tensor(8) tensor(9)
2369 tensor(8) tensor(5)
2387 tensor(1) tensor(9)
2395 tensor(3) tensor(8)
2408 tensor(9) tensor(3)
2414 tensor(4) tensor(9)
2422 tensor(4) tensor(6)
2450 tensor(8) tensor(3)
2462 tensor(0) tensor(2)
2488 tensor(4) tensor(2)
2597 tensor(3) tensor(5)
2648 tensor(5) tensor(9)
2654 tensor(1) tensor(6)
2720 tensor(4) tensor(9)
2723 tensor(8) tensor(3)
2743 tensor(8) tensor(5)
2863 tensor(4) tensor(9)
2877 tensor(7) tensor(4)
2921 tensor(2) tensor(3)
2927 tensor(2) tensor(3)
2930 tensor(3) tensor(5)
2939 tensor(5) tensor(9)
2953 tensor(8) tensor(3)
2995 tensor(8) tensor(6)
3030 tensor(8) tensor(6)
```

```
3073 tensor(2) tensor(1)
3078 tensor(8) tensor(3)
3115 tensor(4) tensor(5)
3117 tensor(4) tensor(5)
3172 tensor(9) tensor(4)
3225 tensor(9) tensor(7)
3330 tensor(8) tensor(2)
3451 tensor(9) tensor(7)
3475 tensor(7) tensor(3)
3503 tensor(1) tensor(9)
3520 tensor(4) tensor(6)
3558 tensor(0) tensor(5)
3565 tensor(6) tensor(5)
3567 tensor(5) tensor(8)
3604 tensor(0) tensor(7)
3702 tensor(4) tensor(5)
3762 tensor(8) tensor(6)
3767 tensor(3) tensor(7)
3776 tensor(8) tensor(5)
3780 tensor(6) tensor(4)
3796 tensor(8) tensor(2)
3808 tensor(3) tensor(7)
3811 tensor(9) tensor(2)
3818 tensor(6) tensor(0)
3853 tensor(2) tensor(6)
3893 tensor(3) tensor(5)
3902 tensor(3) tensor(5)
3906 tensor(3) tensor(1)
3929 tensor(6) tensor(5)
3941 tensor(6) tensor(4)
3943 tensor(5) tensor(3)
3946 tensor(8) tensor(2)
3984 tensor(4) tensor(9)
3985 tensor(4) tensor(9)
4000 tensor(8) tensor(9)
4007 tensor(8) tensor(7)
4027 tensor(1) tensor(7)
4053 tensor(4) tensor(7)
4065 tensor(8) tensor(0)
4078 tensor(7) tensor(9)
4154 tensor(4) tensor(9)
4199 tensor(9) tensor(7)
4201 tensor(7) tensor(1)
4248 tensor(8) tensor(2)
4261 tensor(6) tensor(5)
4289 tensor(8) tensor(2)
4360 tensor(3) tensor(5)
4369 tensor(4) tensor(9)
4443 tensor(2) tensor(3)
4497 tensor(3) tensor(8)
4498 tensor(4) tensor(7)
4500 tensor(1) tensor(9)
4536 tensor(5) tensor(6)
4540 tensor(3) tensor(7)
4548 tensor(6) tensor(5)
4551 tensor(5) tensor(7)
4571 tensor(8) tensor(6)
4615 tensor(4) tensor(2)
4619 tensor(8) tensor(6)
4635 tensor(5) tensor(3)
```

```
4740 tensor(5) tensor(3)
4751 tensor(6) tensor(4)
4761 tensor(4) tensor(9)
4763 tensor(6) tensor(5)
4798 tensor(8) tensor(6)
4807 tensor(3) tensor(8)
4823 tensor(4) tensor(9)
4860 tensor(9) tensor(4)
4880 tensor(8) tensor(0)
4915 tensor(6) tensor(5)
4966 tensor(4) tensor(7)
5457 tensor(8) tensor(1)
5642 tensor(5) tensor(1)
5649 tensor(9) tensor(7)
5654 tensor(3) tensor(7)
5676 tensor(3) tensor(4)
5734 tensor(2) tensor(3)
5821 tensor(3) tensor(5)
5887 tensor(5) tensor(7)
5888 tensor(2) tensor(4)
5913 tensor(3) tensor(5)
5937 tensor(3) tensor(5)
5955 tensor(8) tensor(3)
5972 tensor(3) tensor(5)
5973 tensor(8) tensor(3)
6011 tensor(9) tensor(3)
6023 tensor(8) tensor(3)
6045 tensor(5) tensor(3)
6059 tensor(5) tensor(3)
6115 tensor(8) tensor(1)
6194 tensor(9) tensor(3)
6243 tensor(4) tensor(7)
6555 tensor(9) tensor(8)
6568 tensor(4) tensor(9)
6571 tensor(7) tensor(9)
6576 tensor(4) tensor(7)
6597 tensor(9) tensor(0)
6651 tensor(8) tensor(0)
6741 tensor(9) tensor(7)
6755 tensor(7) tensor(8)
6783 tensor(6) tensor(1)
6847 tensor(4) tensor(6)
7216 tensor(6) tensor(0)
7268 tensor(4) tensor(7)
7434 tensor(8) tensor(4)
7472 tensor(9) tensor(2)
7565 tensor(4) tensor(7)
7574 tensor(1) tensor(4)
7811 tensor(8) tensor(1)
7899 tensor(8) tensor(1)
7921 tensor(2) tensor(8)
7928 tensor(8) tensor(1)
7934 tensor(8) tensor(1)
7990 tensor(8) tensor(1)
8061 tensor(9) tensor(4)
8062 tensor(8) tensor(5)
8091 tensor(8) tensor(2)
8094 tensor(8) tensor(2)
8277 tensor(8) tensor(3)
8279 tensor(4) tensor(8)
```

```
8325 tensor(3) tensor(0)
8339 tensor(2) tensor(8)
8375 tensor(4) tensor(7)
8453 tensor(3) tensor(5)
8527 tensor(9) tensor(4)
8607 tensor(8) tensor(3)
9009 tensor(2) tensor(7)
9015 tensor(2) tensor(7)
9019 tensor(2) tensor(7)
9024 tensor(2) tensor(7)
9163 tensor(2) tensor(3)
9422 tensor(3) tensor(5)
9427 tensor(3) tensor(5)
9517 tensor(4) tensor(9)
9540 tensor(8) tensor(1)
9587 tensor(4) tensor(9)
9634 tensor(1) tensor(0)
9664 tensor(7) tensor(2)
9679 tensor(2) tensor(6)
9698 tensor(3) tensor(6)
9700 tensor(8) tensor(2)
9709 tensor(6) tensor(5)
9719 tensor(0) tensor(5)
9729 tensor(6) tensor(5)
9745 tensor(2) tensor(4)
9749 tensor(6) tensor(5)
9764 tensor(9) tensor(4)
9768 tensor(0) tensor(2)
9770 tensor(0) tensor(5)
9781 tensor(9) tensor(7)
9793 tensor(5) tensor(6)
9839 tensor(3) tensor(2)
9858 tensor(8) tensor(6)
9867 tensor(8) tensor(2)
9913 tensor(3) tensor(2)
9916 tensor(9) tensor(7)
9941 tensor(8) tensor(5)
9944 tensor(8) tensor(3)
9970 tensor(3) tensor(5)
9980 tensor(7) tensor(2)
length: 277
```

In [28]:

```python
def model_C():
    model = torch.nn.Sequential(
        torch.nn.Linear(28*28, 512),
        torch.nn.ReLU(),
        torch.nn.Linear(512, 256),
        torch.nn.ReLU(),
        torch.nn.Linear(256, 128),
        torch.nn.ReLU(),
        torch.nn.Linear(128, 64),
        torch.nn.ReLU(),
        torch.nn.Linear(64, 10)
    )

    # Use Adam as optimizer.
    opt = torch.optim.Adam(params=model.parameters(), lr=0.005)

    # Use CrossEntropyLoss for as loss function.
    loss_fn = torch.nn.CrossEntropyLoss()
    return model, opt, loss_fn
```

In [ ]:

```
model_C_1500_20 = train_model(mnist_training, model_B(), 1500, 20)
```

```
Epoch: 0, Loss: 0.16302679479122162
Epoch: 1, Loss: 0.1140366643667221
Epoch: 2, Loss: 0.08282681554555893
Epoch: 3, Loss: 0.052643027156591415
Epoch: 4, Loss: 0.033914435654878616
Epoch: 5, Loss: 0.040588926523923874
Epoch: 6, Loss: 0.03866032510995865
Epoch: 7, Loss: 0.021335123106837273
Epoch: 8, Loss: 0.01833491027355194
Epoch: 9, Loss: 0.023928174749016762
Epoch: 10, Loss: 0.022946802899241447
Epoch: 11, Loss: 0.015144268050789833
Epoch: 12, Loss: 0.022060032933950424
Epoch: 13, Loss: 0.03052649460732937
Epoch: 14, Loss: 0.023171020671725273
Epoch: 15, Loss: 0.017144806683063507
Epoch: 16, Loss: 0.0068892440758645535
Epoch: 17, Loss: 0.008720414713025093
Epoch: 18, Loss: 0.008278501220047474
Epoch: 19, Loss: 0.004837760701775551
```

In [29]:

```
model_C_2000_30 = train_model(mnist_training, model_B(), 2000, 30)
```

```
Epoch: 0, Loss: 0.24869896471500397
Epoch: 1, Loss: 0.14163294434547424
Epoch: 2, Loss: 0.09183985739946365
Epoch: 3, Loss: 0.056221239268779755
Epoch: 4, Loss: 0.05370711162686348
Epoch: 5, Loss: 0.03146413713693619
Epoch: 6, Loss: 0.03311198204755783
Epoch: 7, Loss: 0.0269969180226326
Epoch: 8, Loss: 0.020287275314331055
Epoch: 9, Loss: 0.01560135930776596
Epoch: 10, Loss: 0.012133071199059486
Epoch: 11, Loss: 0.008829738944768906
Epoch: 12, Loss: 0.012539075687527657
Epoch: 13, Loss: 0.01251035463064909
Epoch: 14, Loss: 0.016222363337874413
Epoch: 15, Loss: 0.0315493568778038
Epoch: 16, Loss: 0.019069252535700798
Epoch: 17, Loss: 0.005301946774125099
Epoch: 18, Loss: 0.009278097189962864
Epoch: 19, Loss: 0.006242542993277311
Epoch: 20, Loss: 0.0034746178425848484
Epoch: 21, Loss: 0.01912214793264866
Epoch: 22, Loss: 0.012941534630954266
Epoch: 23, Loss: 0.009055327624082565
Epoch: 24, Loss: 0.009695162065327168
Epoch: 25, Loss: 0.015208118595182896
Epoch: 26, Loss: 0.006070659030228853
Epoch: 27, Loss: 0.0025385518092662096
Epoch: 28, Loss: 0.011813536286354065
Epoch: 29, Loss: 0.0067063309252262115
```

In [30]:

```python
pred_classes, accuracy = test_model(data, labels, model_C_2000_30)
#print(pred_classes)

errors = []

print(" i, prediction, label")
for n in range(10000):
    if (pred_classes[n] != labels[n]):
        print(n, pred_classes[n], labels[n])
        errors.append(n)

print("length:",errors.__len__())
```

```
0.9798
 i, prediction, label
115 tensor(9) tensor(4)
149 tensor(3) tensor(2)
247 tensor(2) tensor(4)
274 tensor(3) tensor(9)
321 tensor(7) tensor(2)
340 tensor(3) tensor(5)
362 tensor(7) tensor(2)
445 tensor(0) tensor(6)
449 tensor(5) tensor(3)
495 tensor(0) tensor(8)
543 tensor(3) tensor(8)
582 tensor(2) tensor(8)
591 tensor(2) tensor(8)
610 tensor(2) tensor(4)
659 tensor(1) tensor(2)
691 tensor(4) tensor(8)
844 tensor(7) tensor(8)
882 tensor(7) tensor(9)
900 tensor(3) tensor(1)
938 tensor(5) tensor(3)
947 tensor(9) tensor(8)
951 tensor(4) tensor(5)
1014 tensor(5) tensor(6)
1039 tensor(2) tensor(7)
1044 tensor(8) tensor(6)
1112 tensor(6) tensor(4)
1181 tensor(1) tensor(6)
1182 tensor(5) tensor(6)
1217 tensor(1) tensor(9)
1226 tensor(2) tensor(7)
1232 tensor(4) tensor(9)
1242 tensor(9) tensor(4)
1247 tensor(5) tensor(9)
1260 tensor(1) tensor(7)
1289 tensor(9) tensor(5)
1299 tensor(7) tensor(5)
1319 tensor(3) tensor(8)
1328 tensor(9) tensor(7)
1393 tensor(3) tensor(5)
1425 tensor(4) tensor(8)
1500 tensor(1) tensor(7)
1522 tensor(9) tensor(7)
1530 tensor(7) tensor(8)
1549 tensor(6) tensor(4)
1569 tensor(4) tensor(6)
1678 tensor(7) tensor(2)
1681 tensor(7) tensor(3)
1721 tensor(1) tensor(7)
1722 tensor(7) tensor(2)
1800 tensor(4) tensor(6)
1878 tensor(3) tensor(8)
1901 tensor(4) tensor(9)
1955 tensor(2) tensor(8)
2024 tensor(9) tensor(7)
2070 tensor(1) tensor(7)
2098 tensor(0) tensor(2)
2109 tensor(7) tensor(3)
2118 tensor(0) tensor(6)
```

```
2130 tensor(9) tensor(4)
2135 tensor(1) tensor(6)
2291 tensor(3) tensor(5)
2293 tensor(4) tensor(9)
2326 tensor(5) tensor(0)
2369 tensor(3) tensor(5)
2387 tensor(1) tensor(9)
2406 tensor(4) tensor(9)
2422 tensor(4) tensor(6)
2455 tensor(2) tensor(0)
2488 tensor(4) tensor(2)
2578 tensor(2) tensor(7)
2597 tensor(3) tensor(5)
2648 tensor(0) tensor(9)
2654 tensor(1) tensor(6)
2720 tensor(4) tensor(9)
2877 tensor(7) tensor(4)
2921 tensor(2) tensor(3)
2927 tensor(2) tensor(3)
2939 tensor(5) tensor(9)
2953 tensor(5) tensor(3)
2995 tensor(5) tensor(6)
3030 tensor(8) tensor(6)
3073 tensor(2) tensor(1)
3172 tensor(9) tensor(4)
3422 tensor(0) tensor(6)
3451 tensor(9) tensor(7)
3490 tensor(9) tensor(4)
3503 tensor(1) tensor(9)
3520 tensor(4) tensor(6)
3533 tensor(5) tensor(4)
3549 tensor(2) tensor(3)
3558 tensor(0) tensor(5)
3567 tensor(5) tensor(8)
3580 tensor(1) tensor(7)
3597 tensor(3) tensor(9)
3681 tensor(8) tensor(2)
3762 tensor(5) tensor(6)
3776 tensor(8) tensor(5)
3780 tensor(6) tensor(4)
3808 tensor(8) tensor(7)
3838 tensor(1) tensor(7)
3853 tensor(5) tensor(6)
3893 tensor(6) tensor(5)
3902 tensor(3) tensor(5)
3941 tensor(2) tensor(4)
3943 tensor(5) tensor(3)
4007 tensor(4) tensor(7)
4065 tensor(3) tensor(0)
4078 tensor(3) tensor(9)
4176 tensor(7) tensor(2)
4199 tensor(9) tensor(7)
4248 tensor(8) tensor(2)
4271 tensor(3) tensor(5)
4289 tensor(7) tensor(2)
4360 tensor(3) tensor(5)
4369 tensor(4) tensor(9)
4433 tensor(1) tensor(7)
4437 tensor(2) tensor(3)
4497 tensor(7) tensor(8)
```

```
4500 tensor(1) tensor(9)
4536 tensor(5) tensor(6)
4547 tensor(2) tensor(6)
4551 tensor(4) tensor(7)
4571 tensor(8) tensor(6)
4601 tensor(4) tensor(8)
4639 tensor(9) tensor(8)
4731 tensor(7) tensor(8)
4740 tensor(5) tensor(3)
4761 tensor(7) tensor(9)
4807 tensor(3) tensor(8)
4823 tensor(4) tensor(9)
4874 tensor(7) tensor(9)
4880 tensor(5) tensor(0)
5140 tensor(5) tensor(3)
5331 tensor(6) tensor(1)
5586 tensor(0) tensor(8)
5642 tensor(5) tensor(1)
5676 tensor(7) tensor(4)
5719 tensor(7) tensor(9)
5745 tensor(1) tensor(7)
5749 tensor(5) tensor(8)
5750 tensor(9) tensor(0)
5757 tensor(7) tensor(9)
5936 tensor(9) tensor(4)
5937 tensor(3) tensor(5)
5955 tensor(8) tensor(3)
5973 tensor(8) tensor(3)
5981 tensor(9) tensor(5)
5982 tensor(3) tensor(5)
6004 tensor(3) tensor(8)
6046 tensor(8) tensor(3)
6166 tensor(3) tensor(9)
6347 tensor(2) tensor(8)
6555 tensor(9) tensor(8)
6571 tensor(3) tensor(9)
6574 tensor(6) tensor(2)
6576 tensor(1) tensor(7)
6597 tensor(7) tensor(0)
6625 tensor(4) tensor(8)
6632 tensor(5) tensor(9)
6651 tensor(8) tensor(0)
6755 tensor(7) tensor(8)
6783 tensor(6) tensor(1)
6817 tensor(4) tensor(9)
6847 tensor(4) tensor(6)
7216 tensor(3) tensor(0)
7713 tensor(5) tensor(8)
7800 tensor(2) tensor(3)
7921 tensor(2) tensor(8)
8062 tensor(8) tensor(5)
8065 tensor(1) tensor(8)
8277 tensor(9) tensor(3)
8279 tensor(4) tensor(8)
8311 tensor(4) tensor(6)
8325 tensor(6) tensor(0)
8408 tensor(5) tensor(8)
8456 tensor(0) tensor(8)
8522 tensor(6) tensor(8)
8527 tensor(9) tensor(4)
```

```
9009 tensor(2) tensor(7)
9015 tensor(2) tensor(7)
9024 tensor(2) tensor(7)
9280 tensor(5) tensor(8)
9427 tensor(3) tensor(5)
9530 tensor(8) tensor(9)
9587 tensor(4) tensor(9)
9634 tensor(8) tensor(0)
9664 tensor(7) tensor(2)
9669 tensor(5) tensor(4)
9679 tensor(3) tensor(6)
9700 tensor(8) tensor(2)
9729 tensor(6) tensor(5)
9742 tensor(8) tensor(3)
9744 tensor(1) tensor(8)
9749 tensor(6) tensor(5)
9755 tensor(5) tensor(8)
9770 tensor(0) tensor(5)
9782 tensor(5) tensor(6)
9808 tensor(4) tensor(9)
9839 tensor(7) tensor(2)
9856 tensor(5) tensor(9)
9888 tensor(0) tensor(6)
9904 tensor(0) tensor(2)
length: 202
```