A preface: Everything is done in 10 or 20 epochs because of the lack of time. I appologize for that in advance.

Task 1

In [ ]:

```
import tensorflow as tf
from tensorflow import keras
from keras import layers
import numpy as np
import matplotlib.pyplot as plt

@tf.function
def contrastive_loss(y_true, y_pred):
    # ...
    return

# Build a simple CNN with strided convolution layers
def define_model():
    inputs = tf.keras.Input(shape=(28,28,1),name='Inputs')
    x = layers.Conv2D(16,kernel_size=(5,5),activation='relu',padding='same',strides=1,name='L1')(inputs)
    x = layers.Conv2D(16,kernel_size=(3,3),activation='relu',padding='same',strides=2,name='L2')(x)
    x = layers.Conv2D(16,kernel_size=(3,3),activation='relu',padding='same',strides=1,name='L3')(x)
    x = layers.Conv2D(16,kernel_size=(3,3),activation='relu',padding='same',strides=2,name='L4')(x)
    x = layers.Flatten()(x)
    embedding_layer = layers.Dense(2, activation='relu', name='Embedding')(x)
    outputs = layers.Dense(10,activation='softmax')(embedding_layer)
    model = keras.Model(inputs=inputs, outputs=outputs)
    return model


def mnist_data():
    (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
    y_train = tf.keras.utils.to_categorical(y_train)
    y_test = tf.keras.utils.to_categorical(y_test)
    x_train = x_train/255.0
    x_test = x_test/255.0
    return x_train, y_train, x_test, y_test
```

```
# Train the model on MNIST data using standard cross-entropy loss
def train_model(model, x_train, y_train, x_test, y_test, epochs=100):
    model.compile(optimizer='adam', loss=['categorical_crossentropy', 'mean_squared_error'], metrics=['accuracy'])
    model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=epochs, batch_size=64)


x_train, y_train, x_test, y_test = mnist_data()
model = define_model()
train_model(model, x_train, y_train, x_test, y_test, 20)
```

```
Epoch 1/20
938/938 [==============================] - 11s 11ms/step - loss: 1.5153 - accuracy: 0.4342 - val_loss: 1.0730 - val_a
ccuracy: 0.5841
Epoch 2/20
938/938 [==============================] - 10s 11ms/step - loss: 0.9894 - accuracy: 0.6128 - val_loss: 0.8748 - val_a
ccuracy: 0.6397
Epoch 3/20
938/938 [==============================] - 10s 10ms/step - loss: 0.8123 - accuracy: 0.7004 - val_loss: 0.6751 - val_a
ccuracy: 0.7732
Epoch 4/20
938/938 [==============================] - 10s 10ms/step - loss: 0.6431 - accuracy: 0.7936 - val_loss: 0.5802 - val_a
ccuracy: 0.8065
Epoch 5/20
938/938 [==============================] - 10s 10ms/step - loss: 0.5552 - accuracy: 0.8138 - val_loss: 0.5394 - val_a
ccuracy: 0.8160
Epoch 6/20
938/938 [==============================] - 10s 10ms/step - loss: 0.5019 - accuracy: 0.8242 - val_loss: 0.4668 - val_a
ccuracy: 0.8330
Epoch 7/20
938/938 [==============================] - 10s 10ms/step - loss: 0.4648 - accuracy: 0.8326 - val_loss: 0.4486 - val_a
ccuracy: 0.8376
Epoch 8/20
938/938 [==============================] - 10s 11ms/step - loss: 0.4362 - accuracy: 0.8378 - val_loss: 0.4158 - val_a
ccuracy: 0.8424
Epoch 9/20
938/938 [==============================] - 10s 10ms/step - loss: 0.4102 - accuracy: 0.8433 - val_loss: 0.3991 - val_a
ccuracy: 0.8442
Epoch 10/20
938/938 [==============================] - 10s 11ms/step - loss: 0.3911 - accuracy: 0.8460 - val_loss: 0.3891 - val_a
ccuracy: 0.8461
Epoch 11/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3761 - accuracy: 0.8498 - val_loss: 0.3615 - val_a
ccuracy: 0.8507
Epoch 12/20
938/938 [==============================] - 10s 11ms/step - loss: 0.3624 - accuracy: 0.8516 - val_loss: 0.3770 - val_a
ccuracy: 0.8507
Epoch 13/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3508 - accuracy: 0.8542 - val_loss: 0.3598 - val_a
ccuracy: 0.8503
Epoch 14/20
938/938 [==============================] - 9s 10ms/step - loss: 0.3405 - accuracy: 0.8567 - val_loss: 0.3670 - val_ac
curacy: 0.8484
```

```
Epoch 15/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3331 - accuracy: 0.8582 - val_loss: 0.3320 - val_a
ccuracy: 0.8580
Epoch 16/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3249 - accuracy: 0.8599 - val_loss: 0.3296 - val_a
ccuracy: 0.8583
Epoch 17/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3206 - accuracy: 0.8605 - val_loss: 0.3309 - val_a
ccuracy: 0.8578
Epoch 18/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3138 - accuracy: 0.8617 - val_loss: 0.3378 - val_a
ccuracy: 0.8533
Epoch 19/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3072 - accuracy: 0.8623 - val_loss: 0.3219 - val_a
ccuracy: 0.8590
Epoch 20/20
938/938 [==============================] - 10s 10ms/step - loss: 0.3013 - accuracy: 0.8631 - val_loss: 0.3188 - val_a
ccuracy: 0.8596
```

In [ ]:
```
model_for_embeddings = tf.keras.Model(inputs=model.input,
                                      outputs=model.get_layer('Embedding').output)
```

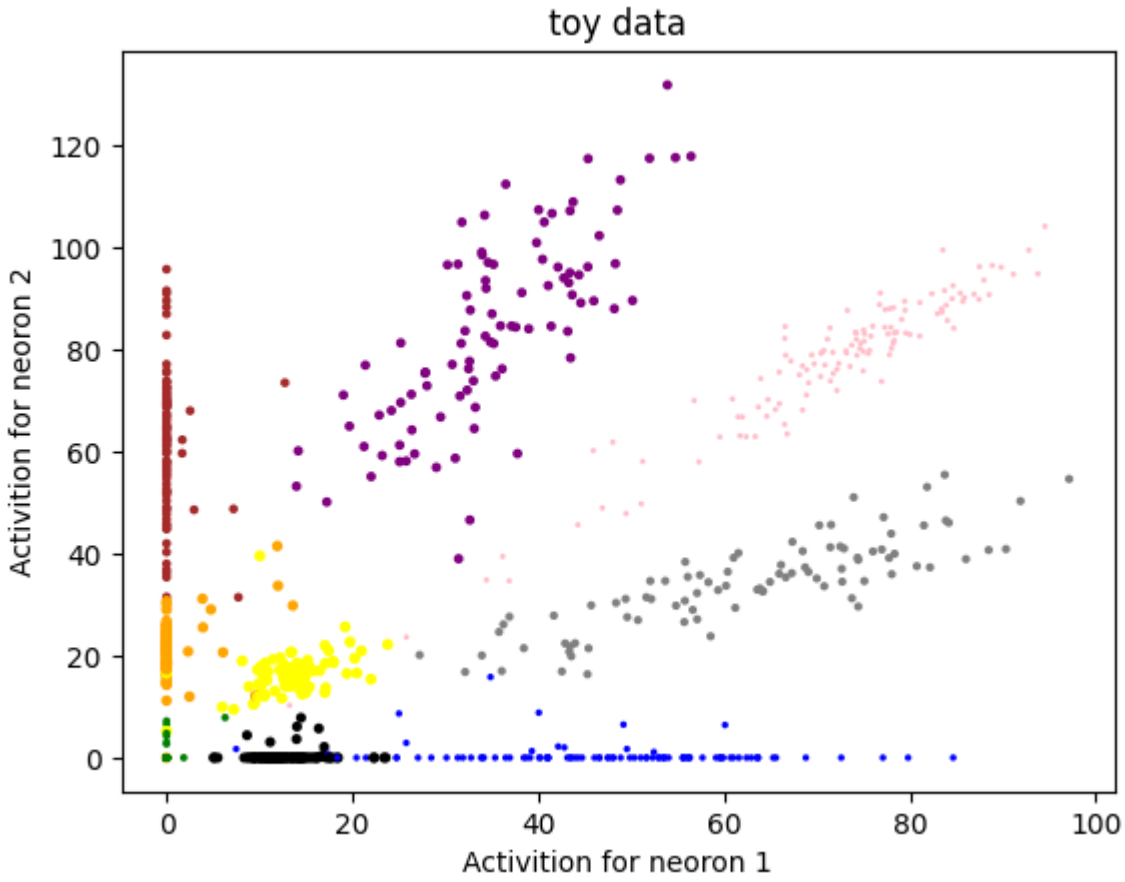
In [ ]:
```
ins_indices = np.random.choice(x_train.shape[0], size=1000, replace=False)
ins = x_train[ins_indices]
labels = y_train[ins_indices].argmax(axis=1)
outs = model_for_embeddings.predict(ins)

colors = {
    0: 'red',
    1: 'pink',
    2: 'blue',
    3: 'green',
    4: 'grey',
    5: 'brown',
    6: 'purple',
    7: 'black',
    8: 'orange',
    9: 'yellow',
}

%matplotlib inline
fig, ax = plt.subplots()
```

```
ax.scatter(outs[:,0], outs[:,1], labels, c=[colors[x] for x in labels])
ax.set_xlabel('Activition for neoron 1')
ax.set_ylabel('Activition for neoron 2')
ax.set_title('toy data')
plt.show()
```

```
32/32 [==============================] - 0s 5ms/step
```



Task 2

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
from keras import layers
```

```python
import numpy as np
import matplotlib.pyplot as plt


batch_size = 8
alpha = 1
learning_rate = 0.0001

@tf.function
def contrastive_loss(y_true, y_pred):
    L = 0.0
    for i in range(batch_size):
        for j in range(i+1, batch_size):
            D = tf.square(y_pred[i] - y_pred[j])
            if y_true[i] == y_true[j]:
                L = L+D
            else:
                L = L+tf.maximum(0.0, alpha-D)
    return L


def mnist_data():
    (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
    y_train = tf.keras.utils.to_categorical(y_train)
    y_test = tf.keras.utils.to_categorical(y_test)
    x_train = x_train/255.0
    x_test = x_test/255.0
    return x_train, y_train, x_test, y_test

# Build a simple CNN with strided convolution layers
def define_model2():
    inputs = tf.keras.Input(shape=(28,28,1),name='Inputs')
    x = layers.Conv2D(16,kernel_size=(5,5),activation='relu',padding='same',strides=1,name='L1')(inputs)
    x = layers.Conv2D(16,kernel_size=(3,3),activation='relu',padding='same',strides=2,name='L2')(x)
    x = layers.Conv2D(16,kernel_size=(3,3),activation='relu',padding='same',strides=1,name='L3')(x)
    x = layers.Conv2D(16,kernel_size=(3,3),activation='relu',padding='same',strides=2,name='L4')(x)
    x = layers.Flatten()(x)
    embedding_layer = layers.Dense(2, activation='relu', name='Embedding')(x)
    model = keras.Model(inputs=inputs, outputs=embedding_layer)
    return model

x_train, y_train, x_test, y_test = mnist_data()
```

```
y_train = y_train.argmax(axis=1)
y_test = y_test.argmax(axis=1)

# Train the model on MNIST data using standard cross-entropy loss
def train_model2(model, x_train, y_train, x_test, y_test, epochs=100):
    model.compile(optimizer=tf.keras.optimizers.Adam(), loss=contrastive_loss, metrics=['accuracy'])
    model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=epochs, batch_size=batch_size)


model2 = define_model2()
train_model2(model2, x_train, y_train, x_test, y_test, 10)
```

```
WARNING:tensorflow:From h:\Uni\WiSe 2024\ML LAB\ml_lab_venv\Lib\site-packages\keras\src\losses.py:2976: The name tf.l
osses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instea
d.

WARNING:tensorflow:From h:\Uni\WiSe 2024\ML LAB\ml_lab_venv\Lib\site-packages\keras\src\backend.py:1398: The name tf.
executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instea
d.

Epoch 1/10
WARNING:tensorflow:From h:\Uni\WiSe 2024\ML LAB\ml_lab_venv\Lib\site-packages\keras\src\utils\tf_utils.py:492: The na
me tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From h:\Uni\WiSe 2024\ML LAB\ml_lab_venv\Lib\site-packages\keras\src\engine\base_layer_utils.py:38
4: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_f
unctions instead.

7500/7500 [==============================] - 51s 6ms/step - loss: 6.1105 - accuracy: 0.0932 - val_loss: 4.3997 - val_
accuracy: 0.1531
Epoch 2/10
7500/7500 [==============================] - 41s 5ms/step - loss: 4.6291 - accuracy: 0.1059 - val_loss: 4.0385 - val_
accuracy: 0.0903
Epoch 3/10
7500/7500 [==============================] - 41s 5ms/step - loss: 4.2005 - accuracy: 0.0947 - val_loss: 3.7564 - val_
accuracy: 0.1005
Epoch 4/10
7500/7500 [==============================] - 42s 6ms/step - loss: 3.9270 - accuracy: 0.1140 - val_loss: 3.6976 - val_
accuracy: 0.0146
Epoch 5/10
7500/7500 [==============================] - 43s 6ms/step - loss: 3.8392 - accuracy: 0.1034 - val_loss: 3.3087 - val_
accuracy: 0.0576
Epoch 6/10
7500/7500 [==============================] - 42s 6ms/step - loss: 3.6256 - accuracy: 0.1059 - val_loss: 3.2903 - val_
accuracy: 0.1345
Epoch 7/10
7500/7500 [==============================] - 42s 6ms/step - loss: 3.5602 - accuracy: 0.1050 - val_loss: 3.3559 - val_
accuracy: 0.0918
Epoch 8/10
7500/7500 [==============================] - 43s 6ms/step - loss: 3.4348 - accuracy: 0.1060 - val_loss: 3.1810 - val_
accuracy: 0.1027
Epoch 9/10
7500/7500 [==============================] - 42s 6ms/step - loss: 3.4328 - accuracy: 0.1073 - val_loss: 3.0855 - val_
accuracy: 0.0989
```

```
Epoch 10/10
7500/7500 [==============================] - 42s 6ms/step - loss: 3.3715 - accuracy: 0.1049 - val_loss: 3.3146 - val_
accuracy: 0.1274
```

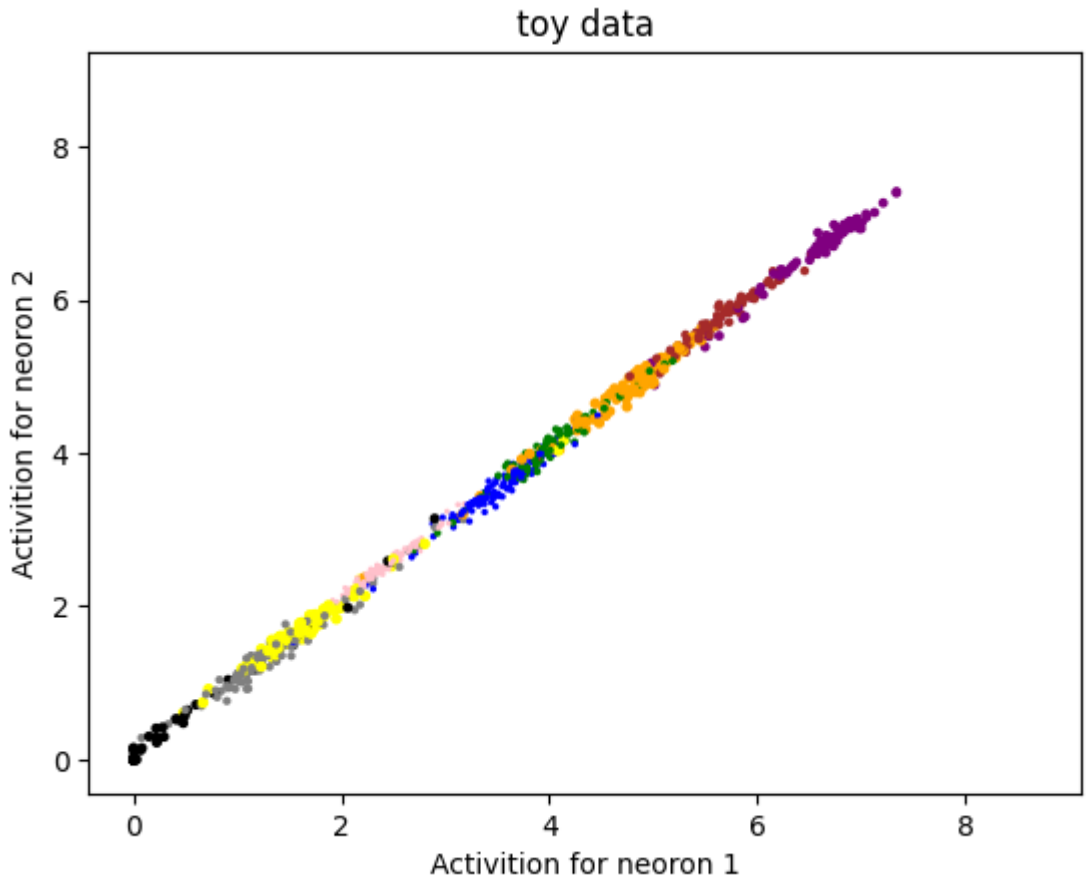
In [ ]:

```python
ins_indices = np.random.choice(x_train.shape[0], size=1000, replace=False)
ins = x_train[ins_indices]
labels = y_train[ins_indices]
outs = model2.predict(ins)

colors = {
    0: 'red',
    1: 'pink',
    2: 'blue',
    3: 'green',
    4: 'grey',
    5: 'brown',
    6: 'purple',
    7: 'black',
    8: 'orange',
    9: 'yellow',
}

%matplotlib inline
fig, ax = plt.subplots()
ax.scatter(outs[:,0], outs[:,1], labels, c=[colors[x] for x in labels])
ax.set_xlabel('Activition for neoron 1')
ax.set_ylabel('Activition for neoron 2')
ax.set_title('toy data')
plt.show()
```

```
32/32 [==============================] - 0s 3ms/step
```


toy data
Activition for neoron 2
Activition for neoron 1
0
2
4
6
8

In [ ]: