A3) Unsupervised Learning with Autoencoders

In [39]:
```
import matplotlib.pyplot as plt
import numpy as np

import requests, pickle, gzip, math
from pathlib import Path

import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F

DATA_PATH = Path("data")
PATH = DATA_PATH / "mnist"
PATH.mkdir(parents=True, exist_ok=True)
URL = "http://deeplearning.net/data/mnist/"
FILENAME = "mnist.pkl.gz"

if not (PATH / FILENAME).exists():
        content = requests.get(URL + FILENAME).content
        (PATH / FILENAME).open("wb").write(content)

with gzip.open((PATH / FILENAME).as_posix(), "rb") as f:
        ((x_train, y_train), (x_valid, y_valid), _) = pickle.load(f, encoding="latin-1")

x_train, y_train, x_valid, y_valid = map(torch.tensor, (x_train, y_train, x_valid, y_valid))

n, c = x_train.shape
```

In [40]:
```
import torch
import torchvision
from torchvision import transforms, datasets
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import matplotlib.pyplot as plt
import numpy as np



train = datasets.MNIST('', train=True, download=True,
                       transform=transforms.Compose([
                           transforms.ToTensor()
                       ]))

test = datasets.MNIST('', train=False, download=True,
                       transform=transforms.Compose([
                           transforms.ToTensor()
                       ]))

trainset = torch.utils.data.DataLoader(train, batch_size=6000, shuffle=False)
testset = torch.utils.data.DataLoader(test, batch_size=6000, shuffle=False)
```

In [43]:
```
global h
h=32
epoch_n=20
```

In [44]:
```
class linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

net = linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)
```

linear_Autoencoder

h=32

In [45]:
```
h=32

class linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

net = linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

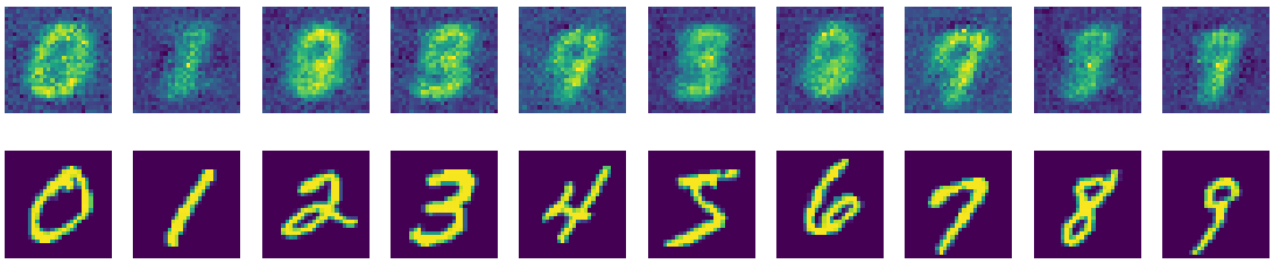
```
tensor(0.1122, grad_fn=<MseLossBackward>)
tensor(0.0914, grad_fn=<MseLossBackward>)
tensor(0.0740, grad_fn=<MseLossBackward>)
tensor(0.0658, grad_fn=<MseLossBackward>)
tensor(0.0624, grad_fn=<MseLossBackward>)
tensor(0.0605, grad_fn=<MseLossBackward>)
tensor(0.0589, grad_fn=<MseLossBackward>)
tensor(0.0571, grad_fn=<MseLossBackward>)
tensor(0.0550, grad_fn=<MseLossBackward>)
tensor(0.0528, grad_fn=<MseLossBackward>)
```

In [46]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 32
```



h=64

In [47]:

```
h=64

class linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

net = linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

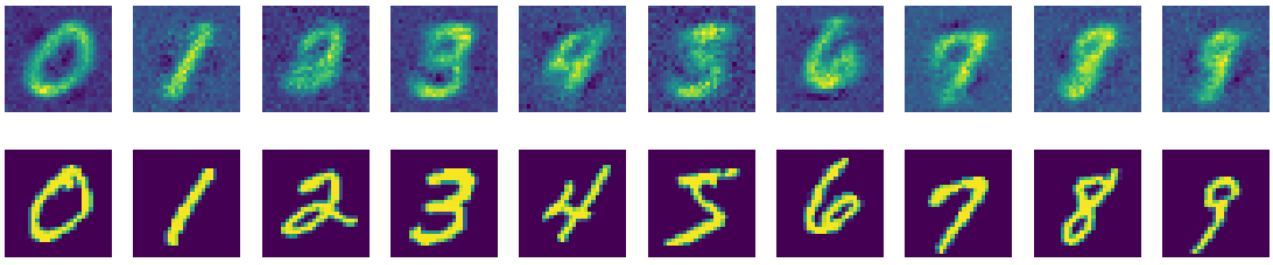
```
tensor(0.0900, grad_fn=<MseLossBackward>)
tensor(0.0656, grad_fn=<MseLossBackward>)
tensor(0.0592, grad_fn=<MseLossBackward>)
tensor(0.0547, grad_fn=<MseLossBackward>)
tensor(0.0505, grad_fn=<MseLossBackward>)
tensor(0.0466, grad_fn=<MseLossBackward>)
tensor(0.0430, grad_fn=<MseLossBackward>)
tensor(0.0400, grad_fn=<MseLossBackward>)
tensor(0.0374, grad_fn=<MseLossBackward>)
tensor(0.0351, grad_fn=<MseLossBackward>)
```

In [48]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 64
```



h=128

In [78]:

```
h=128

class linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

net = linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

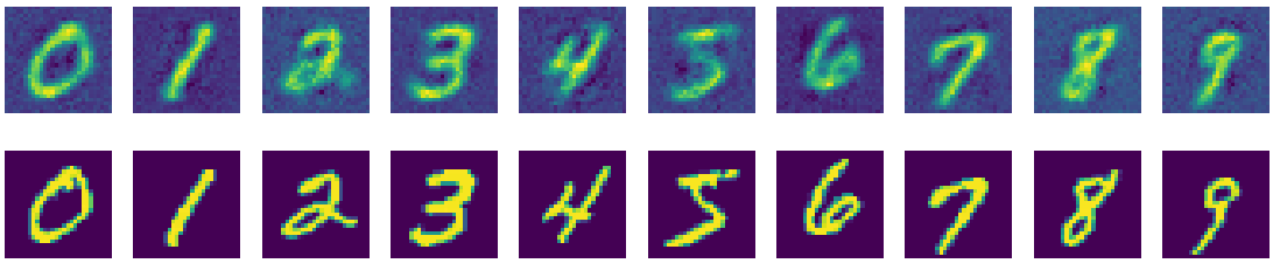
```
tensor(0.0681, grad_fn=<MseLossBackward>)
tensor(0.0556, grad_fn=<MseLossBackward>)
tensor(0.0460, grad_fn=<MseLossBackward>)
tensor(0.0388, grad_fn=<MseLossBackward>)
tensor(0.0339, grad_fn=<MseLossBackward>)
tensor(0.0303, grad_fn=<MseLossBackward>)
tensor(0.0274, grad_fn=<MseLossBackward>)
tensor(0.0249, grad_fn=<MseLossBackward>)
tensor(0.0227, grad_fn=<MseLossBackward>)
tensor(0.0208, grad_fn=<MseLossBackward>)
```

In [79]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 128
```



c) F_1(x) (use h = 128 here) on the test set

In [80]:

```
# I run this code after training with F_1 with h=128

LOSS=0
for data in testset:
    X, y = data
    net.zero_grad()
    output = net(X.view(-1,784))
    LOSS+= loss_function(output, X.view(-1,784))
    #loss.backward()
    #optimizer.step()
print("F_1(x) with h = 128 loss on the test set is", LOSS.detach().numpy())
```

```
F_1(x) with h = 128 loss on the test set is 0.04061942
```

b) Non_linear Autoencoder

In [51]:
```
class non_linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x=F.relu(x)
        x = self.decoder(x)
        x=F.relu(x)
        return x
net = non_linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)
```

h=32

In [52]:
```
h=32

class non_linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x=F.relu(x)
        x = self.decoder(x)
        x=F.relu(x)
        return x

net = non_linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

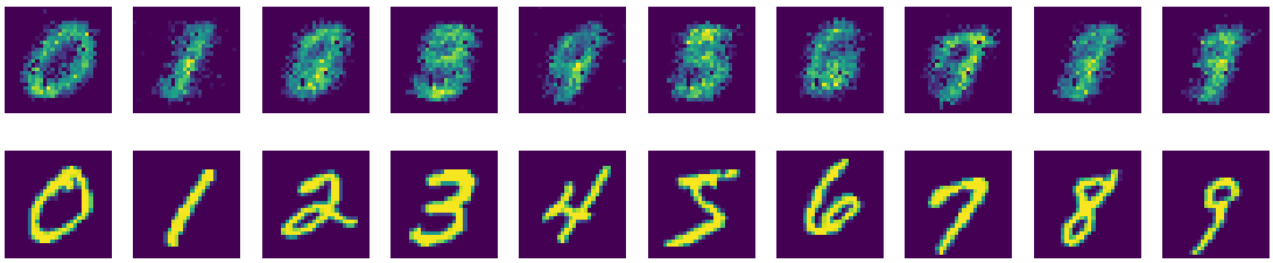
```
tensor(0.0949, grad_fn=<MseLossBackward>)
tensor(0.0842, grad_fn=<MseLossBackward>)
tensor(0.0745, grad_fn=<MseLossBackward>)
tensor(0.0676, grad_fn=<MseLossBackward>)
tensor(0.0631, grad_fn=<MseLossBackward>)
tensor(0.0595, grad_fn=<MseLossBackward>)
tensor(0.0561, grad_fn=<MseLossBackward>)
tensor(0.0529, grad_fn=<MseLossBackward>)
tensor(0.0500, grad_fn=<MseLossBackward>)
tensor(0.0470, grad_fn=<MseLossBackward>)
```

In [53]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 32
```



h=64

In [54]:

```python
h=64

class non_linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x=F.relu(x)
        x = self.decoder(x)
        x=F.relu(x)
        return x

net = non_linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

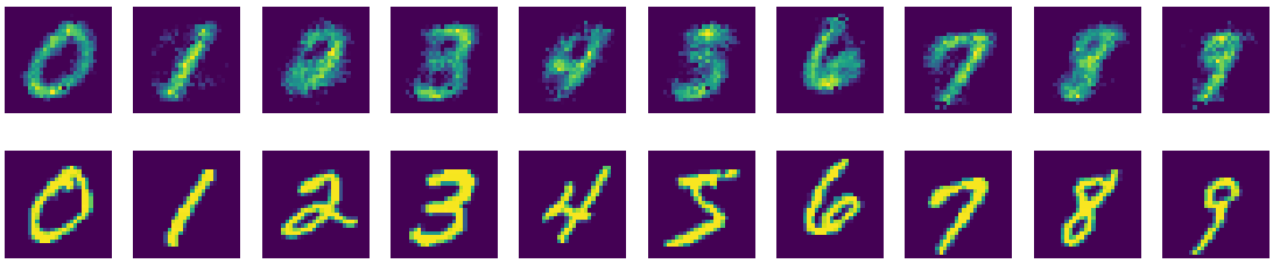
```
tensor(0.0852, grad_fn=<MseLossBackward>)
tensor(0.0688, grad_fn=<MseLossBackward>)
tensor(0.0598, grad_fn=<MseLossBackward>)
tensor(0.0539, grad_fn=<MseLossBackward>)
tensor(0.0482, grad_fn=<MseLossBackward>)
tensor(0.0431, grad_fn=<MseLossBackward>)
tensor(0.0387, grad_fn=<MseLossBackward>)
tensor(0.0350, grad_fn=<MseLossBackward>)
tensor(0.0317, grad_fn=<MseLossBackward>)
tensor(0.0291, grad_fn=<MseLossBackward>)
```

In [55]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 64
```



h=128

In [81]:

```python
h=128

class non_linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x=F.relu(x)
        x = self.decoder(x)
        x=F.relu(x)
        return x

net = non_linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

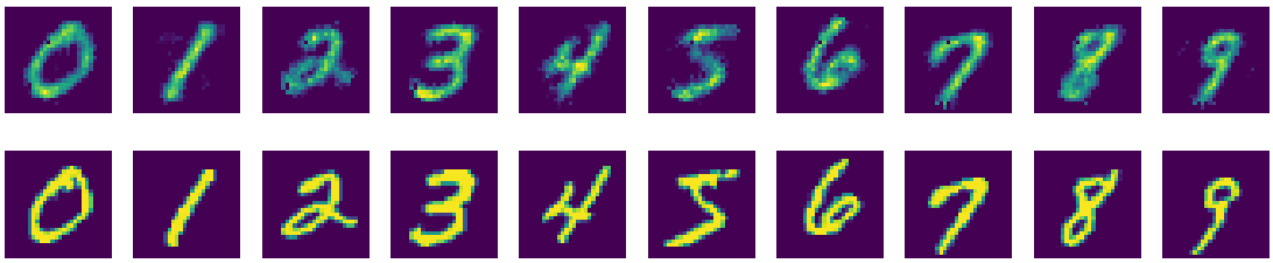
```
tensor(0.0707, grad_fn=<MseLossBackward>)
tensor(0.0561, grad_fn=<MseLossBackward>)
tensor(0.0456, grad_fn=<MseLossBackward>)
tensor(0.0375, grad_fn=<MseLossBackward>)
tensor(0.0313, grad_fn=<MseLossBackward>)
tensor(0.0268, grad_fn=<MseLossBackward>)
tensor(0.0235, grad_fn=<MseLossBackward>)
tensor(0.0207, grad_fn=<MseLossBackward>)
tensor(0.0186, grad_fn=<MseLossBackward>)
tensor(0.0169, grad_fn=<MseLossBackward>)
```

In [82]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 128
```



c) F_2(x) (use h = 128 here) on the test set

In [83]:

```
# I run this code after training with F_2 with h=128

LOSS=0
for data in testset:
    X, y = data
    net.zero_grad()
    output = net(X.view(-1,784))
    LOSS+= loss_function(output, X.view(-1,784))
    #loss.backward()
    #optimizer.step()
print("F_2(x) with h = 128 loss on the test set is", LOSS.detach().numpy())
```

```
F_2(x) with h = 128 loss on the test set is 0.033017926
```

In [ ]:

```

```

out of curiosity part

h=10

In [77]:

```python
h=10

class non_linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x=F.relu(x)
        x = self.decoder(x)
        x=F.relu(x)
        return x

net = non_linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

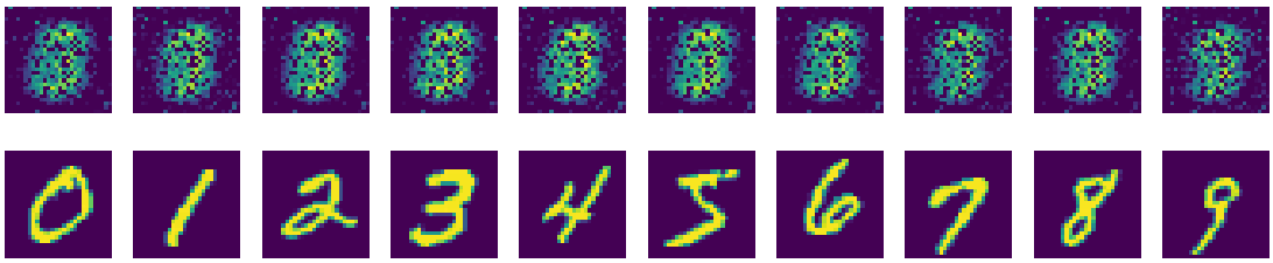
```
tensor(0.1042, grad_fn=<MseLossBackward>)
tensor(0.1011, grad_fn=<MseLossBackward>)
tensor(0.0974, grad_fn=<MseLossBackward>)
tensor(0.0933, grad_fn=<MseLossBackward>)
tensor(0.0889, grad_fn=<MseLossBackward>)
tensor(0.0848, grad_fn=<MseLossBackward>)
tensor(0.0812, grad_fn=<MseLossBackward>)
tensor(0.0784, grad_fn=<MseLossBackward>)
tensor(0.0763, grad_fn=<MseLossBackward>)
tensor(0.0748, grad_fn=<MseLossBackward>)
```

In [59]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 10
```



h=28*28

In [60]:

```
h=28*28

class non_linear_Autoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Linear(28*28, h)
        self.decoder = nn.Linear(h, 28*28)

    def forward(self, x):
        x = self.encoder(x)
        x=F.relu(x)
        x = self.decoder(x)
        x=F.relu(x)
        return x

net = non_linear_Autoencoder()
loss_function = nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

for epoch in range(epoch_n):
    for data in trainset:
        X, y = data
        net.zero_grad()
        output = net(X.view(-1,784))
        loss = loss_function(output, X.view(-1,784))
        loss.backward()
        optimizer.step()
    print(loss)
```

```
tensor(0.0514, grad_fn=<MseLossBackward>)
tensor(0.0317, grad_fn=<MseLossBackward>)
tensor(0.0223, grad_fn=<MseLossBackward>)
tensor(0.0163, grad_fn=<MseLossBackward>)
tensor(0.0126, grad_fn=<MseLossBackward>)
tensor(0.0104, grad_fn=<MseLossBackward>)
tensor(0.0089, grad_fn=<MseLossBackward>)
tensor(0.0079, grad_fn=<MseLossBackward>)
tensor(0.0072, grad_fn=<MseLossBackward>)
tensor(0.0066, grad_fn=<MseLossBackward>)
```

In [61]:

```
i=-1
N=10
index=[1,3,5,7,9,0,13,15,17,19]
plt.figure(figsize=(14,3),dpi=100)
print('h=',h)
for j in range(N):
    i+=1
    j+=1
    plt.subplot(2,N,j)
    ww=net(x_train[index[i]]).detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)
    plt.subplot(2,N,N+j)
    ww=x_train[index[i]].detach().numpy()
    x_shape=np.reshape(ww,(28,28))
    plt.axis('off')
    plt.imshow(x_shape)

plt.show()
```

```
h= 784
```