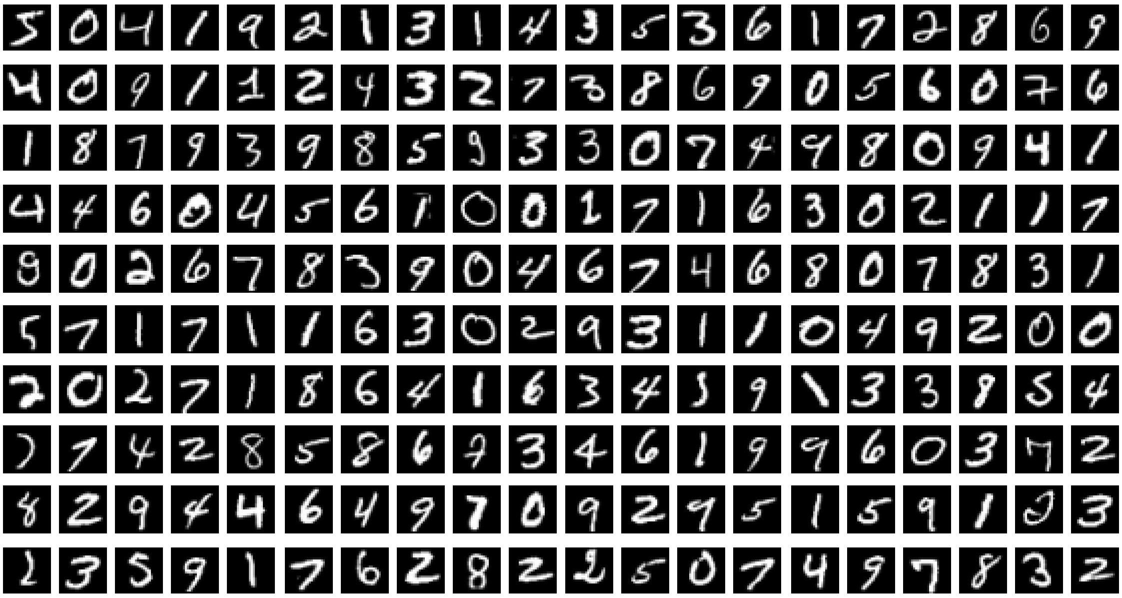
TP 1 - Algorithme de rétro-propagation de l’erreur

In [1]:
```
nom='jai'
prenom='ilyass'
```

In [2]:
```
from keras.datasets import mnist
# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

```
60000 train samples
10000 test samples
```

In [3]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.figure(figsize=(7.195, 3.841), dpi=100)
for i in range(200):
  plt.subplot(10,20,i+1)
  plt.imshow(X_train[i,:].reshape([28,28]), cmap='gray')
  plt.axis('off')
plt.show()
```



In [4]:
```
# print the shape of the data
print(X_train.shape)
```

```
(60000, 784)
```

Question : Quel est l’espace dans lequel se trouvent les images? Quelle est sa taille?

L'espace des images est un espace vectoriel où chaque pixel est un composant du vecteur, les valeurs de chaque pixel sont des nombres compris entre 0 et 1 car les images sont standardisées par la valeur maximale du niveau de gris (255). La taille de cet espace est de 28x28 = 784 pixels. Chacune de ces images est donc représentée par un vecteur de 784 dimensions.

exercice 1 : Régression Logistique:

- Q1: Comme on a une seule couche complètement connectée, le nombre de paramètres du modèles sera le nombre de poids liant chaque neurone d'entrée (784 composantes du vecteur d'entrée) à chaque neurone en sortie (10 classes), s'ajoute à celà le nombre de biais s'ajoutant à chaque sortie, donc : Nb_param = card(Wi)+card(bi) = Nb_input*Nb_output+Nb_output = (784*10)+10 = 7850

- Q2 :

La fonction de coût est définie comme :

$$L_{W,b}(D) = -\frac{1}{N}\sum_{i=1}^{N}\log(\hat{y}_{c^*,i})$$

Où :

$$\log(\hat{y}_{c^*,i}) = \log\left(\frac{e^{\langle \mathbf{x}_i, \mathbf{w}_c\rangle + b_c}}{\sum_{c'=1}^{10} e^{\langle \mathbf{x}_i, \mathbf{w}_{c'}\rangle + b_{c'}}}\right)$$

Cela peut être reformulé comme :

$$\log(\hat{y}_{c^*,i}) = \langle \mathbf{x}_i, \mathbf{w}_c\rangle + b_c - \log\left(\sum_{c'=1}^{10} e^{\langle \mathbf{x}_i, \mathbf{w}_{c'}\rangle + b_{c'}}\right)$$

- Le premier terme est linéaire par rapport à wc et bc, et le second terme est une fonction convexe par rapport à wc et bc car c'est le logarithme de la somme des exponentielles. On peut donc conclure que la fonction de coût est convexe car elle est la somme d'une fonction linéaire et d'une fonction convexe. Et en utilisant la propriété convexe, nous sommes sûrs que nous pouvons converger vers le minimum global de la solution en utilisant un bon pas de gradient.

- Q3: Définition de la fonction softmax :

$$\hat{y}_i = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

Fonction de perte :

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log(\hat{y}_{c^*,i})$$

Pour un seul exemple :

$$L = -\log(\hat{y}_{c^*})$$

Dérivée de (L) par rapport à (\hat{y}_i) :

$$\frac{\partial L}{\partial \hat{y}_i} = \begin{cases} -\frac{1}{\hat{y}_i}, & \text{si } i = c^* \\ 0, & \text{sinon.} \end{cases}$$

Dérivée de (\hat{y}_i) par rapport à (s_i) :

$$\frac{\partial \hat{y}_i}{\partial s_k} = \begin{cases} \hat{y}_i(1-\hat{y}_i), & \text{si } i = k \\ -\hat{y}_i\hat{y}_k, & \text{si } i \neq k \end{cases}$$

Chaînage des dérivées :

$$\frac{\partial L}{\partial s_i} = \sum_k \frac{\partial L}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial s_i}$$

Résultat final :

$$\frac{\partial L}{\partial s_i} = \hat{y}_i - y_i^*$$

- Q4 : En déduire que :

$$\frac{\partial L}{\partial W} = \frac{1}{N}X^T(\hat{Y} - Y^*) = \frac{1}{N}X^T\Delta y$$

$$\frac{\partial L}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i^*)$$

On a que :

$$\frac{\partial s_i}{\partial W} = x_i^T$$

$$\frac{\partial s_i}{\partial b} = I_k$$

Et donc, par application de la "chain rule", on aura :

$$\frac{\partial L}{\partial W} = \frac{1}{N}X^T(\hat{Y} - Y^*) = \frac{1}{N}X^T\Delta y$$

$$\frac{\partial L}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i^*)$$

In [14]:
```
from keras.utils import to_categorical
```

```
K=10 # number of classes
# convert class vectors to binary class matrices
Y_train = to_categorical(y_train, K)
Y_test = to_categorical(y_test, K)
```

For keras > 2.0, please use from keras.utils import to_categorical instead.

Example of usage will be to_categorical(y, num_classes=None)

https://stackoverflow.com/questions/45149341/importerror-cannot-import-name-np-utils

In [15]:
```
import numpy as np
def softmax(X):
    E = np.exp(X)
    return (E.T / np.sum(E, axis=1)).T
```

In [16]:
```
def forward(batch, W, b):
    linear_projection = np.dot(batch, W) + b
    return softmax(linear_projection)
```

In [17]:
```
def compute_gradients(X_batch, Y_batch, Y_pred, batch_size):
    dW = (1 / batch_size) *np.dot(X_batch.T, Y_pred - Y_batch)
    db = (1 / batch_size) *np.sum(Y_pred - Y_batch, axis=0, keepdims=True)
    return dW, db
```

In [18]:
```
def update_parameters(W, b, gradW, gradb, eta):
    W -= eta * gradW
    b -= eta * gradb
    return W, b
```

In [19]:
```
def accuracy(W, b, images, labels):
  pred = forward(images, W,b )
  return np.where( pred.argmax(axis=1) != labels.argmax(axis=1) , 0.,1.).mean()*
```

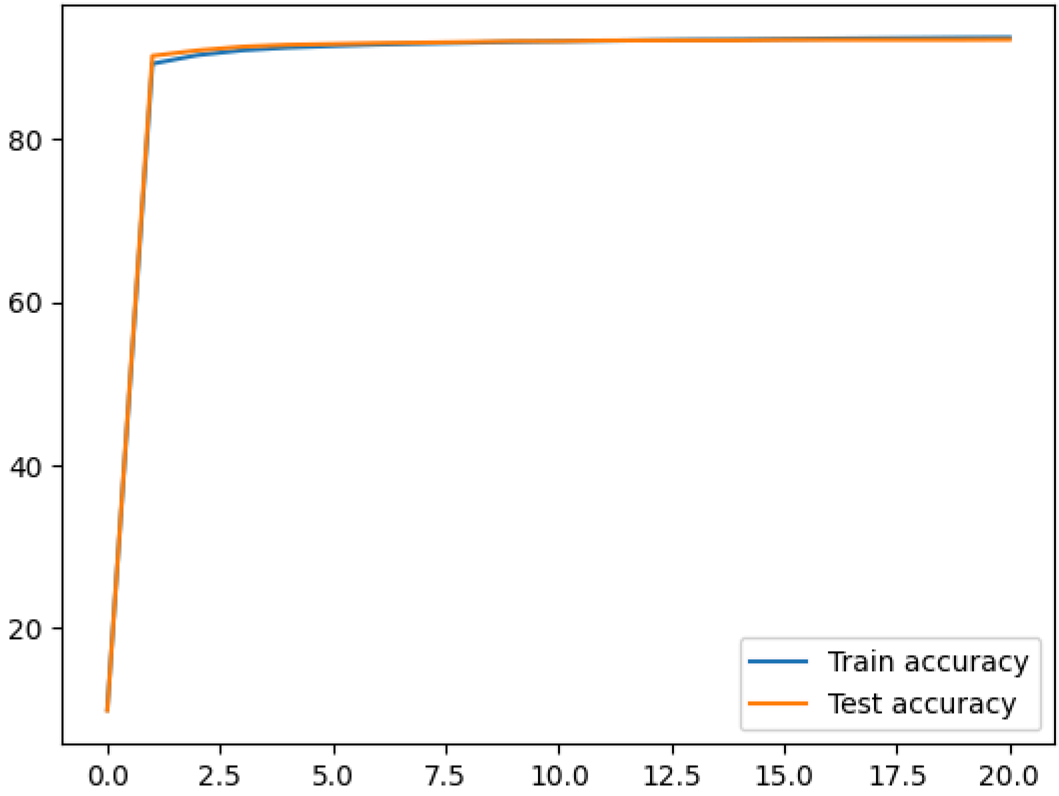
In [20]:
```
N = X_train.shape[0]
d = X_train.shape[1]
W = np.zeros((d,K))
b = np.zeros((1,K))
numEp1 = 20 # Number of epochs for gradient descent
eta1 = 1e-1 # Learning rate
batch_size = 100
nb_batches = int(float(N) / batch_size)

gradW = np.zeros((d,K)) #creation d'un vecteur gradient de W nul
gradb = np.zeros((1,K)) #creation d'un vecteur gradient de b nul

train_accuracy = accuracy(W, b, X_train, Y_train)
test_accuracy = accuracy(W, b, X_test, Y_test)

train_acc = []
test_acc = []

train_acc.append(train_accuracy)
test_acc.append(test_accuracy)

for epoch in range(numEp1):
```

```
    for batch_idx in range(nb_batches):
      start_idx = batch_idx * batch_size
      end_idx = (batch_idx + 1) * batch_size
      X_batch = X_train[start_idx:end_idx, :]
      Y_batch = Y_train[start_idx:end_idx, :]

       # FORWARD PASS : compute prediction with current params for examples in bat
      Y_pred = forward(X_batch, W, b)

       # BACKWARD PASS :
       # 1) compute gradients for W and b
      gradW, gradb = compute_gradients(X_batch, Y_batch, Y_pred, batch_size=batch_
       # 2) update W and b parameters with gradient descent
      W, b = update_parameters(W, b, gradW, gradb, eta1)

    train_accuracy = accuracy(W, b, X_train, Y_train)
    test_accuracy = accuracy(W, b, X_test, Y_test)
    train_acc.append(train_accuracy)
    test_acc.append(test_accuracy)
    print(f"Epoch {epoch + 1}/{numEp1} - Train Accuracy: {train_accuracy:.4f} - Te
```

```
Epoch 1/20 - Train Accuracy: 89.2733 - Test Accuracy: 90.2400
Epoch 2/20 - Train Accuracy: 90.3300 - Test Accuracy: 90.9100
Epoch 3/20 - Train Accuracy: 90.9283 - Test Accuracy: 91.3700
Epoch 4/20 - Train Accuracy: 91.2533 - Test Accuracy: 91.6000
Epoch 5/20 - Train Accuracy: 91.4533 - Test Accuracy: 91.7000
Epoch 6/20 - Train Accuracy: 91.6233 - Test Accuracy: 91.8100
Epoch 7/20 - Train Accuracy: 91.7483 - Test Accuracy: 91.8700
Epoch 8/20 - Train Accuracy: 91.8500 - Test Accuracy: 91.9500
Epoch 9/20 - Train Accuracy: 91.9600 - Test Accuracy: 92.0400
Epoch 10/20 - Train Accuracy: 92.0350 - Test Accuracy: 92.0400
Epoch 11/20 - Train Accuracy: 92.0900 - Test Accuracy: 92.1300
Epoch 12/20 - Train Accuracy: 92.1733 - Test Accuracy: 92.1400
Epoch 13/20 - Train Accuracy: 92.2333 - Test Accuracy: 92.1500
Epoch 14/20 - Train Accuracy: 92.2600 - Test Accuracy: 92.1500
Epoch 15/20 - Train Accuracy: 92.3117 - Test Accuracy: 92.2100
Epoch 16/20 - Train Accuracy: 92.3667 - Test Accuracy: 92.2200
Epoch 17/20 - Train Accuracy: 92.4033 - Test Accuracy: 92.2200
Epoch 18/20 - Train Accuracy: 92.4417 - Test Accuracy: 92.2200
Epoch 19/20 - Train Accuracy: 92.4800 - Test Accuracy: 92.2200
Epoch 20/20 - Train Accuracy: 92.5017 - Test Accuracy: 92.2400
```

In [21]:

```
# Plotting the evolution of the train and test accuracy
plt.plot(train_acc, label='Train accuracy')
plt.plot(test_acc, label='Test accuracy')
plt.legend()
plt.show()
```



exercice 3 : MPL

- Question :cette fonction de coût est-elle convexe par rapport aux paramètres b du modèle? Avec un pas de gradient bien choisi, peut-on assurer la convergence vers le minimum global de la solution?

Dans un modèle de perceptron multicouche (MLP) comportant au moins une couche cachée, la fonction de coût est non convexe. Cela signifie qu'il peut y avoir plusieurs minima locaux dans la fonction, plutôt qu'un seul minimum global. Il peut donc être difficile de trouver le minimum global à l'aide des méthodes d'optimisation traditionnelles, telles que la descente du gradient. Même avec un pas de gradient bien choisi pour l'algorithme de descente de gradient, le modèle peut converger vers un minimum local plutôt que vers le minimum global, ce qui entraîne des performances sous-optimales. De plus, cette non-convexité fait qu'il est difficile de garantir que l'algorithme d'optimisation convergera vers une solution. Il s'agit d'un inconvénient important des MLP et d'autres modèles non convexes, car il limite leur capacité à trouver la solution optimale globale pour le problème à résoudre.

In [27]:
```
def sigmoid(X):
    return 1 / (1 + np.exp(-X))
```

In [28]:
```
def forwardMLP(batch, Wh, bh, Wy, by):
    ui = np.dot(batch, Wh) + bh
    hi = sigmoid(ui)
    vi = np.dot(hi, Wy) + by
    Y_pred = softmax(vi)
```

```
    return Y_pred, hi
```

In [29]:
```
def compute_gradientsMPL(X_batch, Y_batch, Y_pred, Wy, hi):
    tb = X_batch.shape[0]

    # Compute gradients for Wy and by
    delta_y = Y_pred - Y_batch
    grad_Wy = (1 / tb) * np.dot(hi.T, delta_y)
    grad_by = (1 / tb) * np.sum(delta_y, axis=0, keepdims=True)

    # Compute gradients for Wh and bh
    delta_h = np.dot(delta_y, Wy.T) * (hi * (1 - hi))
    grad_Wh = (1 / tb) * np.dot(X_batch.T, delta_h)
    grad_bh = (1 / tb) * np.sum(delta_h, axis=0, keepdims=True)

    return grad_Wh, grad_bh, grad_Wy, grad_by
```

In [30]:
```
def update_parametersMPL(Wh, bh, Wy, by, grad_Wh, grad_bh, grad_Wy, grad_by, eta
    Wh -= eta * grad_Wh
    bh -= eta * grad_bh
    Wy -= eta * grad_Wy
    by -= eta * grad_by

    return Wh, bh, Wy, by
```

In [31]:
```
def accuracyMPL(X, Wh, bh, Wy, by, labels):
  pred, h = forwardMLP(X, Wh, bh, Wy, by)
  return np.where( pred.argmax(axis=1) != labels.argmax(axis=1) , 0.,1.).mean()*
```

In [41]:
```
K = 10
L = 100
numEp2 = 100  # Number of epochs for gradient descent
eta2 = 1  # Learning rate
batch_size = 100
nb_batches = int(float(N) / batch_size)
```

Initialisation avec des zéros uniquement

In [34]:
```
Wh = np.zeros((d, L))
bh = np.zeros((1, L))
Wy = np.zeros((L, K))
by = np.zeros((1, K))

train_accuracy3 = accuracyMPL(X_batch, Wh, bh, Wy, by, Y_batch)
test_accuracy3 = accuracyMPL(X_test, Wh, bh, Wy, by, Y_test)

train_acc3 = []
test_acc3 = []

train_acc3.append(train_accuracy3)
test_acc3.append(test_accuracy3)

for epoch in range(numEp2):
    for batch_idx in range(nb_batches):
        # Select a batch
        start_idx = batch_idx * batch_size
```

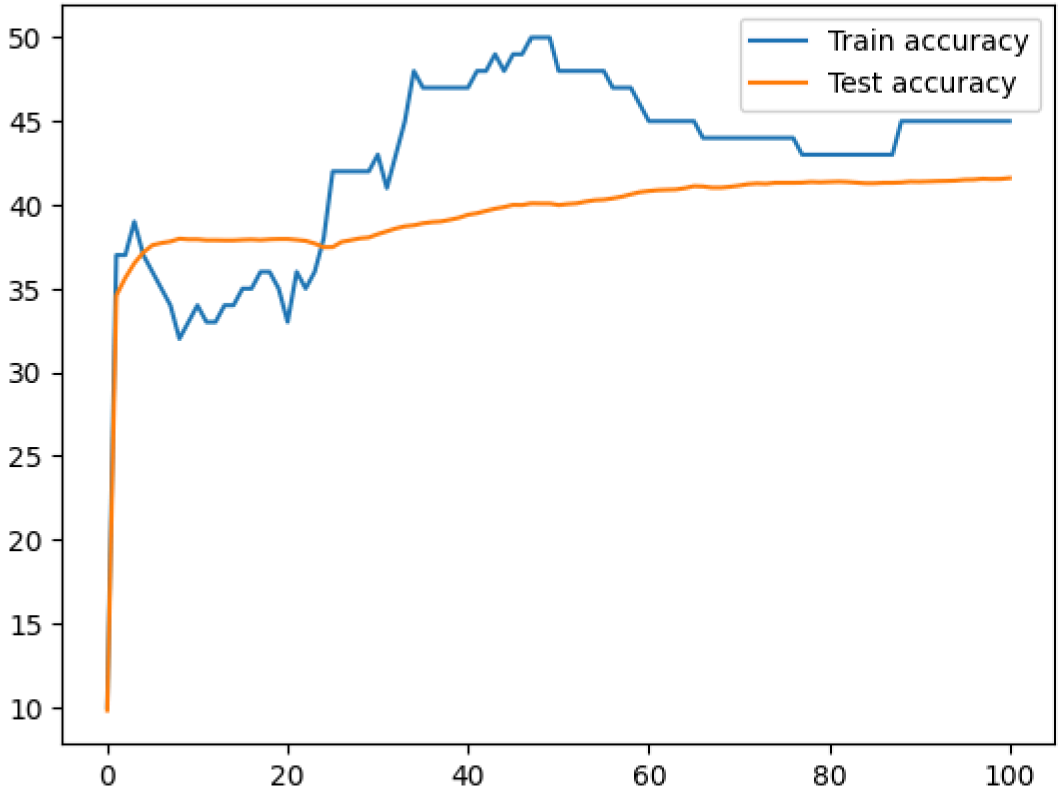
```
        end_idx = (batch_idx + 1) * batch_size
        X_batch = X_train[start_idx:end_idx, :]
        Y_batch = Y_train[start_idx:end_idx, :]

        # Forward pass
        Y_pred, hi = forwardMLP(X_batch, Wh, bh, Wy, by)

        # Backward pass
        grad_Wh, grad_bh, grad_Wy, grad_by = compute_gradientsMPL(X_batch, Y_bat

        # Update parameters
        Wh, bh, Wy, by = update_parametersMPL(Wh, bh, Wy, by, grad_Wh, grad_bh,


    train_accuracy3 = accuracyMPL(X_batch, Wh, bh, Wy, by, Y_batch)
    test_accuracy3 = accuracyMPL(X_test, Wh, bh, Wy, by, Y_test)
    train_acc3.append(train_accuracy3)
    test_acc3.append(test_accuracy3)
    print(f"Epoch {epoch + 1}/{numEp2} - Train Accuracy: {train_accuracy3:.4f} -
```

```
Epoch 1/100 - Train Accuracy: 37.0000 - Test Accuracy: 34.5800
Epoch 2/100 - Train Accuracy: 37.0000 - Test Accuracy: 35.6400
Epoch 3/100 - Train Accuracy: 39.0000 - Test Accuracy: 36.5100
Epoch 4/100 - Train Accuracy: 37.0000 - Test Accuracy: 37.1600
Epoch 5/100 - Train Accuracy: 36.0000 - Test Accuracy: 37.5900
Epoch 6/100 - Train Accuracy: 35.0000 - Test Accuracy: 37.7200
Epoch 7/100 - Train Accuracy: 34.0000 - Test Accuracy: 37.8100
Epoch 8/100 - Train Accuracy: 32.0000 - Test Accuracy: 37.9800
Epoch 9/100 - Train Accuracy: 33.0000 - Test Accuracy: 37.9400
Epoch 10/100 - Train Accuracy: 34.0000 - Test Accuracy: 37.9400
Epoch 11/100 - Train Accuracy: 33.0000 - Test Accuracy: 37.8900
Epoch 12/100 - Train Accuracy: 33.0000 - Test Accuracy: 37.8900
Epoch 13/100 - Train Accuracy: 34.0000 - Test Accuracy: 37.8800
Epoch 14/100 - Train Accuracy: 34.0000 - Test Accuracy: 37.8800
Epoch 15/100 - Train Accuracy: 35.0000 - Test Accuracy: 37.9100
Epoch 16/100 - Train Accuracy: 35.0000 - Test Accuracy: 37.9300
Epoch 17/100 - Train Accuracy: 36.0000 - Test Accuracy: 37.9000
Epoch 18/100 - Train Accuracy: 36.0000 - Test Accuracy: 37.9400
Epoch 19/100 - Train Accuracy: 35.0000 - Test Accuracy: 37.9600
Epoch 20/100 - Train Accuracy: 33.0000 - Test Accuracy: 37.9600
Epoch 21/100 - Train Accuracy: 36.0000 - Test Accuracy: 37.9100
Epoch 22/100 - Train Accuracy: 35.0000 - Test Accuracy: 37.8500
Epoch 23/100 - Train Accuracy: 36.0000 - Test Accuracy: 37.6900
Epoch 24/100 - Train Accuracy: 38.0000 - Test Accuracy: 37.4700
Epoch 25/100 - Train Accuracy: 42.0000 - Test Accuracy: 37.4800
Epoch 26/100 - Train Accuracy: 42.0000 - Test Accuracy: 37.7900
Epoch 27/100 - Train Accuracy: 42.0000 - Test Accuracy: 37.8900
Epoch 28/100 - Train Accuracy: 42.0000 - Test Accuracy: 38.0000
Epoch 29/100 - Train Accuracy: 42.0000 - Test Accuracy: 38.0500
Epoch 30/100 - Train Accuracy: 43.0000 - Test Accuracy: 38.2500
Epoch 31/100 - Train Accuracy: 41.0000 - Test Accuracy: 38.4200
Epoch 32/100 - Train Accuracy: 43.0000 - Test Accuracy: 38.5900
Epoch 33/100 - Train Accuracy: 45.0000 - Test Accuracy: 38.7200
Epoch 34/100 - Train Accuracy: 48.0000 - Test Accuracy: 38.7800
Epoch 35/100 - Train Accuracy: 47.0000 - Test Accuracy: 38.9000
Epoch 36/100 - Train Accuracy: 47.0000 - Test Accuracy: 38.9700
Epoch 37/100 - Train Accuracy: 47.0000 - Test Accuracy: 39.0100
Epoch 38/100 - Train Accuracy: 47.0000 - Test Accuracy: 39.1100
Epoch 39/100 - Train Accuracy: 47.0000 - Test Accuracy: 39.2300
Epoch 40/100 - Train Accuracy: 47.0000 - Test Accuracy: 39.4100
Epoch 41/100 - Train Accuracy: 48.0000 - Test Accuracy: 39.5000
Epoch 42/100 - Train Accuracy: 48.0000 - Test Accuracy: 39.6400
Epoch 43/100 - Train Accuracy: 49.0000 - Test Accuracy: 39.7700
Epoch 44/100 - Train Accuracy: 48.0000 - Test Accuracy: 39.8700
Epoch 45/100 - Train Accuracy: 49.0000 - Test Accuracy: 40.0000
Epoch 46/100 - Train Accuracy: 49.0000 - Test Accuracy: 39.9900
Epoch 47/100 - Train Accuracy: 50.0000 - Test Accuracy: 40.0900
Epoch 48/100 - Train Accuracy: 50.0000 - Test Accuracy: 40.0800
Epoch 49/100 - Train Accuracy: 50.0000 - Test Accuracy: 40.0800
Epoch 50/100 - Train Accuracy: 48.0000 - Test Accuracy: 39.9900
Epoch 51/100 - Train Accuracy: 48.0000 - Test Accuracy: 40.0500
Epoch 52/100 - Train Accuracy: 48.0000 - Test Accuracy: 40.0900
Epoch 53/100 - Train Accuracy: 48.0000 - Test Accuracy: 40.2000
Epoch 54/100 - Train Accuracy: 48.0000 - Test Accuracy: 40.2800
Epoch 55/100 - Train Accuracy: 48.0000 - Test Accuracy: 40.3100
Epoch 56/100 - Train Accuracy: 47.0000 - Test Accuracy: 40.3900
Epoch 57/100 - Train Accuracy: 47.0000 - Test Accuracy: 40.4900
Epoch 58/100 - Train Accuracy: 47.0000 - Test Accuracy: 40.6300
Epoch 59/100 - Train Accuracy: 46.0000 - Test Accuracy: 40.7600
Epoch 60/100 - Train Accuracy: 45.0000 - Test Accuracy: 40.8300
```

```
Epoch 61/100 - Train Accuracy: 45.0000 - Test Accuracy: 40.8800
Epoch 62/100 - Train Accuracy: 45.0000 - Test Accuracy: 40.9000
Epoch 63/100 - Train Accuracy: 45.0000 - Test Accuracy: 40.9200
Epoch 64/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.0000
Epoch 65/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.1200
Epoch 66/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.1000
Epoch 67/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.0300
Epoch 68/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.0300
Epoch 69/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.0800
Epoch 70/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.1500
Epoch 71/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.2400
Epoch 72/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.2800
Epoch 73/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.2600
Epoch 74/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.3200
Epoch 75/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.3200
Epoch 76/100 - Train Accuracy: 44.0000 - Test Accuracy: 41.3200
Epoch 77/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3300
Epoch 78/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3700
Epoch 79/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3500
Epoch 80/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3700
Epoch 81/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3900
Epoch 82/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3700
Epoch 83/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3300
Epoch 84/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.2900
Epoch 85/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.2900
Epoch 86/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3200
Epoch 87/100 - Train Accuracy: 43.0000 - Test Accuracy: 41.3200
Epoch 88/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.3400
Epoch 89/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.3900
Epoch 90/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.3800
Epoch 91/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.4000
Epoch 92/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.4200
Epoch 93/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.4300
Epoch 94/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.4500
Epoch 95/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.5000
Epoch 96/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.5100
Epoch 97/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.5600
Epoch 98/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.5400
Epoch 99/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.5500
Epoch 100/100 - Train Accuracy: 45.0000 - Test Accuracy: 41.6000
```

In [36]:

```python
# Plotting the evolution of the train and test accuracy
plt.plot(train_acc3, label='Train accuracy')
plt.plot(test_acc3, label='Test accuracy')
plt.legend()
plt.show()
```



- La précision d'entraînement reste bloquée à 45 %.
- La précision de test oscille légèrement autour de 41.5 %, sans amélioration significative.
- Cela indique que le réseau n'apprend pas efficacement au fil des époques.

Initialisation avec une distribution normale de faible écart-type (0.1)

In [37]:
```
Wh = np.random.normal(loc=0.0, scale=0.1, size=(d, L))
bh = np.random.normal(loc=0.0, scale=0.1, size=(1, L))
Wy = np.random.normal(loc=0.0, scale=0.1, size=(L, K))
by = np.random.normal(loc=0.0, scale=0.1, size=(1, K))

train_accuracy4 = accuracyMPL(X_batch, Wh, bh, Wy, by, Y_batch)
test_accuracy4 = accuracyMPL(X_test, Wh, bh, Wy, by, Y_test)

train_acc4 = []
test_acc4 = []

train_acc4.append(train_accuracy4)
test_acc4.append(test_accuracy4)

for epoch in range(numEp2):
    for batch_idx in range(nb_batches):
        # Select a batch
        start_idx = batch_idx * batch_size
        end_idx = (batch_idx + 1) * batch_size
        X_batch = X_train[start_idx:end_idx, :]
        Y_batch = Y_train[start_idx:end_idx, :]
```

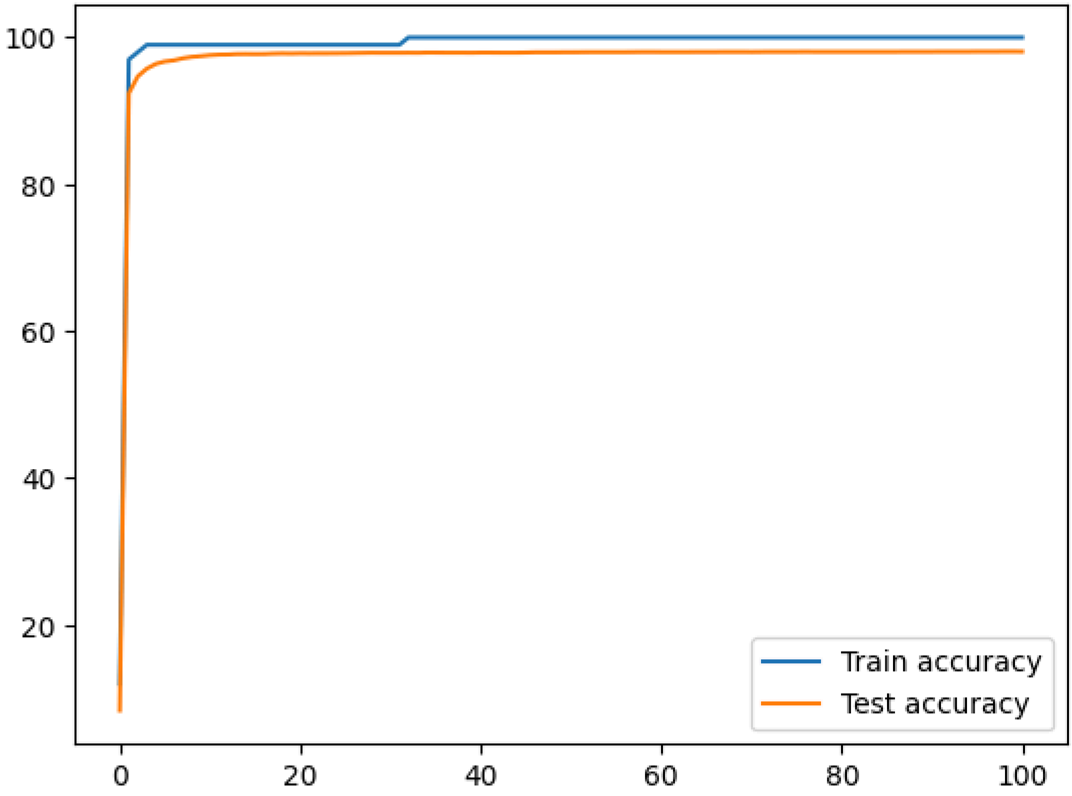
```
        # Forward pass
        Y_pred, hi = forwardMLP(X_batch, Wh, bh, Wy, by)

        # Backward pass
        grad_Wh, grad_bh, grad_Wy, grad_by = compute_gradientsMPL(X_batch, Y_bat

        # Update parameters
        Wh, bh, Wy, by = update_parametersMPL(Wh, bh, Wy, by, grad_Wh, grad_bh,


    train_accuracy4 = accuracyMPL(X_batch, Wh, bh, Wy, by, Y_batch)
    test_accuracy4 = accuracyMPL(X_test, Wh, bh, Wy, by, Y_test)
    train_acc4.append(train_accuracy4)
    test_acc4.append(test_accuracy4)
    print(f"Epoch {epoch + 1}/{numEp2} - Train Accuracy: {train_accuracy4:.4f} -
```

```
Epoch 1/100 - Train Accuracy: 97.0000 - Test Accuracy: 92.3900
Epoch 2/100 - Train Accuracy: 98.0000 - Test Accuracy: 94.7100
Epoch 3/100 - Train Accuracy: 99.0000 - Test Accuracy: 95.7500
Epoch 4/100 - Train Accuracy: 99.0000 - Test Accuracy: 96.4000
Epoch 5/100 - Train Accuracy: 99.0000 - Test Accuracy: 96.7300
Epoch 6/100 - Train Accuracy: 99.0000 - Test Accuracy: 96.8800
Epoch 7/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.1700
Epoch 8/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.3600
Epoch 9/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.4600
Epoch 10/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.5900
Epoch 11/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6600
Epoch 12/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6800
Epoch 13/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7400
Epoch 14/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7500
Epoch 15/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7400
Epoch 16/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7500
Epoch 17/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8000
Epoch 18/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8300
Epoch 19/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8100
Epoch 20/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8400
Epoch 21/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8300
Epoch 22/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8400
Epoch 23/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8400
Epoch 24/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8500
Epoch 25/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8600
Epoch 26/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.8800
Epoch 27/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.9000
Epoch 28/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.9200
Epoch 29/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.9100
Epoch 30/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.9100
Epoch 31/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.9100
Epoch 32/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9200
Epoch 33/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9100
Epoch 34/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9400
Epoch 35/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9400
Epoch 36/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9300
Epoch 37/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9300
Epoch 38/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9300
Epoch 39/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9200
Epoch 40/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9400
Epoch 41/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9400
Epoch 42/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9300
Epoch 43/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9300
Epoch 44/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9300
Epoch 45/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9500
Epoch 46/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9800
Epoch 47/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9800
Epoch 48/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9800
Epoch 49/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9900
Epoch 50/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.9900
Epoch 51/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0000
Epoch 52/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0100
Epoch 53/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0100
Epoch 54/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0100
Epoch 55/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0100
Epoch 56/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0100
Epoch 57/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0200
Epoch 58/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0300
Epoch 59/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0300
Epoch 60/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0200
```

```
Epoch 61/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0200
Epoch 62/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0200
Epoch 63/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0200
Epoch 64/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0300
Epoch 65/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0300
Epoch 66/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0300
Epoch 67/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0400
Epoch 68/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0400
Epoch 69/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0400
Epoch 70/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0300
Epoch 71/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0400
Epoch 72/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 73/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 74/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 75/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 76/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0600
Epoch 77/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 78/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 79/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 80/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 81/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 82/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 83/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 84/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 85/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 86/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0500
Epoch 87/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0600
Epoch 88/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0600
Epoch 89/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0600
Epoch 90/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0600
Epoch 91/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0700
Epoch 92/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0700
Epoch 93/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0700
Epoch 94/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0700
Epoch 95/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0800
Epoch 96/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0800
Epoch 97/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0800
Epoch 98/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0900
Epoch 99/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0900
Epoch 100/100 - Train Accuracy: 100.0000 - Test Accuracy: 98.0800
```

In [38]:

```
# Plotting the evolution of the train and test accuracy
plt.plot(train_acc4, label='Train accuracy')
plt.plot(test_acc4, label='Test accuracy')
plt.legend()
plt.show()
```



Initialisation avec normalisation (valeurs réduites par 748)

In [39]:

```
Wh = np.random.normal(loc=0.0, scale=0.1, size=(d, L))/748
bh = np.random.normal(loc=0.0, scale=0.1, size=(1, L))/748
Wy = np.random.normal(loc=0.0, scale=0.1, size=(L, K))/748
by = np.random.normal(loc=0.0, scale=0.1, size=(1, K))/748

train_accuracy5 = accuracyMPL(X_batch, Wh, bh, Wy, by, Y_batch)
test_accuracy5 = accuracyMPL(X_test, Wh, bh, Wy, by, Y_test)

train_acc5 = []
test_acc5 = []

train_acc5.append(train_accuracy5)
test_acc5.append(test_accuracy5)

for epoch in range(numEp2):
    for batch_idx in range(nb_batches):
        # Select a batch
        start_idx = batch_idx * batch_size
        end_idx = (batch_idx + 1) * batch_size
        X_batch = X_train[start_idx:end_idx, :]
        Y_batch = Y_train[start_idx:end_idx, :]

        # Forward pass
        Y_pred, hi = forwardMLP(X_batch, Wh, bh, Wy, by)

        # Backward pass
        grad_Wh, grad_bh, grad_Wy, grad_by = compute_gradientsMPL(X_batch, Y_bat

        # Update parameters
        Wh, bh, Wy, by = update_parametersMPL(Wh, bh, Wy, by, grad_Wh, grad_bh,
```

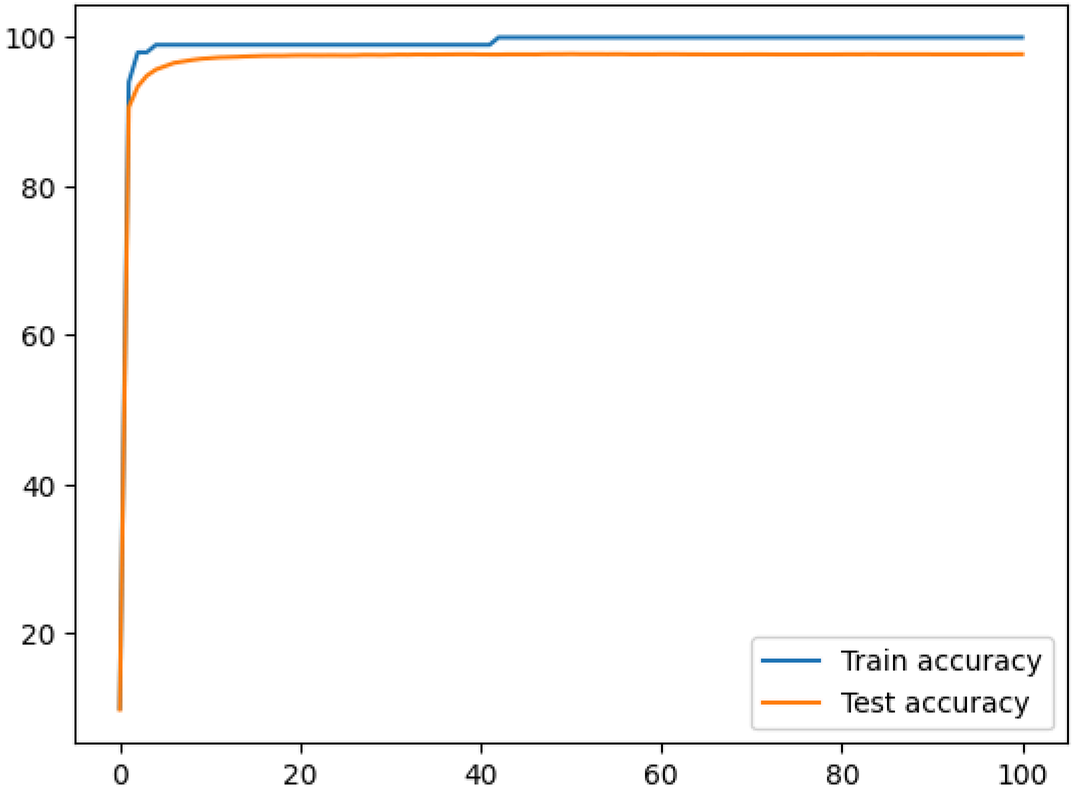
```
    train_accuracy5 = accuracyMPL(X_batch, Wh, bh, Wy, by, Y_batch)
    test_accuracy5 = accuracyMPL(X_test, Wh, bh, Wy, by, Y_test)
    train_acc5.append(train_accuracy5)
    test_acc5.append(test_accuracy5)
    print(f"Epoch {epoch + 1}/{numEp2} - Train Accuracy: {train_accuracy5:.4f} -
```

```
Epoch 1/100 - Train Accuracy: 94.0000 - Test Accuracy: 90.6300
Epoch 2/100 - Train Accuracy: 98.0000 - Test Accuracy: 93.4200
Epoch 3/100 - Train Accuracy: 98.0000 - Test Accuracy: 94.8600
Epoch 4/100 - Train Accuracy: 99.0000 - Test Accuracy: 95.6800
Epoch 5/100 - Train Accuracy: 99.0000 - Test Accuracy: 96.1300
Epoch 6/100 - Train Accuracy: 99.0000 - Test Accuracy: 96.5700
Epoch 7/100 - Train Accuracy: 99.0000 - Test Accuracy: 96.7700
Epoch 8/100 - Train Accuracy: 99.0000 - Test Accuracy: 96.9700
Epoch 9/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.1200
Epoch 10/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.2200
Epoch 11/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.3100
Epoch 12/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.3500
Epoch 13/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.3900
Epoch 14/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.4500
Epoch 15/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.4900
Epoch 16/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.5300
Epoch 17/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.5400
Epoch 18/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.5400
Epoch 19/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.5800
Epoch 20/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6000
Epoch 21/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6000
Epoch 22/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.5900
Epoch 23/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6000
Epoch 24/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6000
Epoch 25/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.5900
Epoch 26/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6000
Epoch 27/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6500
Epoch 28/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6500
Epoch 29/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6300
Epoch 30/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6600
Epoch 31/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6900
Epoch 32/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.6800
Epoch 33/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7100
Epoch 34/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7000
Epoch 35/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7000
Epoch 36/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7200
Epoch 37/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7300
Epoch 38/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7400
Epoch 39/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7400
Epoch 40/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7100
Epoch 41/100 - Train Accuracy: 99.0000 - Test Accuracy: 97.7000
Epoch 42/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7000
Epoch 43/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 44/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 45/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 46/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 47/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7500
Epoch 48/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7600
Epoch 49/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7600
Epoch 50/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7800
Epoch 51/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7600
Epoch 52/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7600
Epoch 53/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7600
Epoch 54/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7500
Epoch 55/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7600
Epoch 56/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7500
Epoch 57/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 58/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 59/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 60/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7500
```

```
Epoch 61/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7500
Epoch 62/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 63/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 64/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 65/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 66/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 67/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 68/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 69/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 70/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 71/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 72/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 73/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7100
Epoch 74/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7100
Epoch 75/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7100
Epoch 76/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7100
Epoch 77/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 78/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 79/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7200
Epoch 80/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 81/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 82/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 83/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7500
Epoch 84/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 85/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 86/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 87/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 88/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 89/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 90/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 91/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 92/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 93/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 94/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 95/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 96/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7300
Epoch 97/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 98/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 99/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7400
Epoch 100/100 - Train Accuracy: 100.0000 - Test Accuracy: 97.7500
```

In [40]:

```python
# Plotting the evolution of the train and test accuracy
plt.plot(train_acc5, label='Train accuracy')
plt.plot(test_acc5, label='Test accuracy')
plt.legend()
plt.show()
```



L'initialisation des poids à zéro entraîne de mauvaises performances dans notre modèle, car les entrées de toutes les couches, à l'exception de la première, seront nulles lorsqu'elles seront multipliées par ces poids, ce qui donnera des sorties identiques. De plus, si on se retrouve proche d'un minimum local , il est très difficile d'en resortir, il y a donc un effet de plateau. Pour améliorer les performances, il est important d'initialiser les poids avec des valeurs différentes. L'uti lisation d'une distribution normale ou de l'initialisation de Xavier pour l'initialisation des poids entraîne une amélioration significative, avec des performances similaires pour les deux méthodes . De plus, l'utilisation d'un perceptron multicouche améliore les performances par rapport à un modèle de régression logistique pour ce problème non-linéairement séparable.
L'initialisation des poids à zéro entraîne de mauvaises performances dans notre modèle, car les entrées de toutes les couches, à l'exception de la première, seront nulles lorsqu'elles seront multipliées par ces poids, ce qui donnera des sorties identiques. De plus, si on se retrouve proche d'un minimum local , il est très difficile d'en resortir, il y a donc un effet de plateau. Pour améliorer les performances, il est important d'initialiser les poids avec des valeurs différentes. L'uti lisation d'une distribution normale ou de l'initialisation de Xavier pour l'initialisation des poids entraîne une amélioration significative, avec des performances similaires pour les deux méthodes . De plus, l'utilisation d'un perceptron multicouche améliore les performances par rapport à un modèle de régression logistique pour ce problème non-linéairement séparable.