```
# Importar las librerías necesarias
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os

# Parámetros del entorno Q-Learning (aunque no se utiliza en este código directamente)
BOARD_SIZE = 25
ACTIONS = ['UP', 'DOWN', 'LEFT', 'RIGHT']
Q = np.zeros((BOARD_SIZE, BOARD_SIZE, len(ACTIONS)))
learning_rate = 0.1
discount_factor = 0.9
epsilon = 0.2
episodes = 1000

# Cargar el dataset Fashion MNIST
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# Normalizar los datos
train_images = train_images / 255.0
test_images = test_images / 255.0

# Definir clases de prendas (solo usaremos 0 y 1)
class_names = [
    'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
    'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'
]

# Filtrar solo las clases 0 (T-shirt/top) y 1 (Trouser)
filter_classes = [0, 1]
train_filter = np.isin(train_labels, filter_classes)
test_filter = np.isin(test_labels, filter_classes)

train_images_filtered = train_images[train_filter]
train_labels_filtered = train_labels[train_filter]
test_images_filtered = test_images[test_filter]
test_labels_filtered = test_labels[test_filter]

# Limitar los datos a 500 ejemplos para inducir sobreajuste
train_images_filtered = train_images_filtered[:500]
train_labels_filtered = train_labels_filtered[:500]

# Crear el modelo
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(128, activation='relu'),
    layers.Dense(10)  # Aun tiene 10 salidas, aunque solo entrenamos 2 clases
])

# Compilar
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Entrenar (sobreajuste inducido con muchas épocas)
model.fit(train_images_filtered, train_labels_filtered, epochs=50)

# Evaluar
test_loss, test_acc = model.evaluate(test_images_filtered, test_labels_filtered, verbose=2)
print(f"\nAccuracy con clases filtradas: {test_acc * 100:.2f}%")

# Función para cargar y preprocesar imágenes desde carpeta
def load_and_preprocess_images_from_folder(folder_path):
    images = []
    for filename in os.listdir(folder_path):
        if filename.endswith('.png') or filename.endswith('.jpg'):
            file_path = os.path.join(folder_path, filename)
            img = Image.open(file_path).convert('L')  # Escala de grises
            img = img.resize((28, 28))  # Redimensionar
            img_array = np.array(img) / 255.0
            images.append(img_array)
    return np.array(images)

# Función para agregar ruido
def add_noise(images, noise_factor=0.5):
    noisy_images = images + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=images.shape)
    noisy_images = np.clip(noisy_images, 0., 1.)
    return noisy_images
```

```
# Cargar imágenes externas desde carpeta 'test'
folder_path = './test'  # Ajusta a tu ruta real
test_images_folder = load_and_preprocess_images_from_folder(folder_path)

# Agregar ruido a propósito
test_images_folder_noisy = add_noise(test_images_folder)

# Predecir
predictions = model.predict(test_images_folder_noisy)

# Mostrar imágenes y predicciones
plt.figure(figsize=(10, 10))
for i in range(min(25, len(test_images_folder_noisy))):
    plt.subplot(5, 5, i + 1)
    plt.imshow(test_images_folder_noisy[i], cmap=plt.cm.binary)
    predicted_label = np.argmax(predictions[i])
    plt.title(f"Pred: {class_names[predicted_label]}")
    plt.xticks([])
    plt.yticks([])
plt.tight_layout()
plt.show()
```

```
Epoch 1/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.6905 - loss: 1.1311
Epoch 2/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9580 - loss: 0.1120
Epoch 3/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9751 - loss: 0.0690
Epoch 4/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9851 - loss: 0.0530
Epoch 5/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9768 - loss: 0.0575
Epoch 6/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9845 - loss: 0.0458
Epoch 7/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9973 - loss: 0.0309
Epoch 8/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9891 - loss: 0.0373
Epoch 9/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9943 - loss: 0.0272
Epoch 10/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9877 - loss: 0.0294
Epoch 11/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9987 - loss: 0.0214
Epoch 12/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9989 - loss: 0.0234
Epoch 13/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9965 - loss: 0.0199
Epoch 14/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9937 - loss: 0.0215
Epoch 15/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9955 - loss: 0.0162
Epoch 16/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9937 - loss: 0.0230
Epoch 17/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9994 - loss: 0.0140
Epoch 18/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9989 - loss: 0.0114
Epoch 19/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0090
Epoch 20/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0082
Epoch 21/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0077
Epoch 22/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0075
Epoch 23/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0085
Epoch 24/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0060
Epoch 25/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0039
Epoch 26/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0066
Epoch 27/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0042
Epoch 28/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0048
Epoch 29/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0039
Epoch 30/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 1.0000 - loss: 0.0038
Epoch 31/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0026
Epoch 32/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0037
Epoch 33/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0041
Epoch 34/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0024
Epoch 35/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0029
Epoch 36/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0024
Epoch 37/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0029
Epoch 38/50
```