**Instalação das dependências –** Codigo tem início com as instalações das dependências como o pytorch e o torchvisio caso eles ainda não estejam instalados, dependendo de onde estamos utilizando esse passo pode não ser necessário.

**Importação de bibliotecas –** Logo após realizamos a importação das bibliotecas necessárias para criar e treinar uma rede neural, também define a utilização da GPU caso disponível.

**Carregamento do dataset MNIST –** Usa o torchvision para baixar e transformar as imagens para tensores e divide o dataset em treino e em test.

**Tornar o dataset iterável –** Define o tamanho do batch e o número total de iterações, logo após, calcula o número de épocas com base no tamanho do dataset e no batch. Realiza a criação dos iteradores ”train\_loader” e “test\_loader” para permitir o carregamento dos batchs de forma aleatória para treino e sequencial para teste

**Definição do modelo CNN –** Define a classe do CNNModel que herda de nn.Module, definindo todos os parâmetros como podemos observar no código.

**Definição da função de custo e do otimizador –** Define a função de custo como CrossEntropyLoss, que aplica o softmax seguido da função -ln(), ideal para problema de classificação multiclasse. Também define o otimizador SGD do torch com taxa de aprendizado igual a 0.01 e peso de decaimento igual a 0.

**Treinamento e avaliação do modelo –** Itera pelo número de épocas exibindo o processo de cada época, para cada batch envia as imagens e rótulos para GPU ou CPU, zera os gradientes, realiza a passagem direta para calcular a saída do modelo, calcula a perda e realiza a passagem para trás para otimização, por último atualiza os parâmetros do modelo. A cada 500 iterações, testa o modelo no conjunto de teste para calcular a precisão, para isso desativa o cálculo de gradiente durante a avaliação e compara as previsões com os rótulos verdadeiros e exibe a perda e o tempo de processamento para cada batch.