Fashion Forward is a new AI-based e-commerce clothing retailer. They want to use image classification to automatically categorize new product listings, making it easier for customers to find what they're looking for. It will also assist in inventory management by quickly sorting items.

As a data scientist tasked with implementing a garment classifier, your primary objective is to develop a machine learning model capable of accurately categorizing images of clothing items into distinct garment types such as shirts, trousers, shoes, etc.

In [1]:
```
# Run the cells below first
```

In [2]:
```
!pip install torchmetrics
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Collecting torchmetrics
  Downloading torchmetrics-1.3.0.post0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: numpy>1.20.0 in /usr/local/lib/python3.8/di
st-packages (from torchmetrics) (1.23.2)
Requirement already satisfied: packaging>17.1 in /usr/local/lib/python3.8/
dist-packages (from torchmetrics) (21.3)
Requirement already satisfied: torch>=1.10.0 in /usr/local/lib/python3.8/d
ist-packages (from torchmetrics) (1.13.0)
Collecting lightning-utilities>=0.8.0 (from torchmetrics)
  Downloading lightning_utilities-0.10.1-py3-none-any.whl.metadata (4.8 kB
)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3
.8/dist-packages (from torchmetrics) (4.5.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist
-packages (from lightning-utilities>=0.8.0->torchmetrics) (65.6.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/
python3.8/dist-packages (from packaging>17.1->torchmetrics) (3.0.9)
Requirement already satisfied: nvidia-cuda-runtime-cu11==11.7.99 in /usr/l
ocal/lib/python3.8/dist-packages (from torch>=1.10.0->torchmetrics) (11.7.
99)
Requirement already satisfied: nvidia-cudnn-cu11==8.5.0.96 in /usr/local/l
ib/python3.8/dist-packages (from torch>=1.10.0->torchmetrics) (8.5.0.96)
Requirement already satisfied: nvidia-cublas-cu11==11.10.3.66 in /usr/loca
l/lib/python3.8/dist-packages (from torch>=1.10.0->torchmetrics) (11.10.3.
66)
Requirement already satisfied: nvidia-cuda-nvrtc-cu11==11.7.99 in /usr/loc
al/lib/python3.8/dist-packages (from torch>=1.10.0->torchmetrics) (11.7.99
)
Requirement already satisfied: wheel in /usr/local/lib/python3.8/dist-pack
ages (from nvidia-cublas-cu11==11.10.3.66->torch>=1.10.0->torchmetrics) (0
.38.4)
Downloading torchmetrics-1.3.0.post0-py3-none-any.whl (840 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 840.2/840.2 kB 18.0 MB/s eta 0
:00:00a 0:00:01
Downloading lightning_utilities-0.10.1-py3-none-any.whl (24 kB)
Installing collected packages: lightning-utilities, torchmetrics
Successfully installed lightning-utilities-0.10.1 torchmetrics-1.3.0.post0
```

In [3]:
```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from torchmetrics import Accuracy, Precision, Recall
```

In [4]:
```
# Load datasets
from torchvision import datasets
import torchvision.transforms as transforms

train_data = datasets.FashionMNIST(root='./data', train=True, download=Tr
test_data = datasets.FashionMNIST(root='./data', train=False, download=Tr
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/tra
in-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/tra
in-images-idx3-ubyte.gz to ./data/FashionMNIST/raw/train-images-idx3-ubyte
.gz
  0%|          | 0/26421880 [00:00<?, ?it/s]
Extracting ./data/FashionMNIST/raw/train-images-idx3-ubyte.gz to ./data/Fa
shionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/tra
in-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/tra
in-labels-idx1-ubyte.gz to ./data/FashionMNIST/raw/train-labels-idx1-ubyte
.gz
  0%|          | 0/29515 [00:00<?, ?it/s]
Extracting ./data/FashionMNIST/raw/train-labels-idx1-ubyte.gz to ./data/Fa
shionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10
k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10
k-images-idx3-ubyte.gz to ./data/FashionMNIST/raw/t10k-images-idx3-ubyte.g
z
  0%|          | 0/4422102 [00:00<?, ?it/s]
Extracting ./data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to ./data/Fas
hionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10
k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10
k-labels-idx1-ubyte.gz to ./data/FashionMNIST/raw/t10k-labels-idx1-ubyte.g
z
  0%|          | 0/5148 [00:00<?, ?it/s]
Extracting ./data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/Fas
hionMNIST/raw
```

In [5]:

```
# Get the number of classes
classes = train_data.classes
num_classes = len(train_data.classes)

# Define some relevant variables
num_input_channels = 1
num_output_channels = 16
image_size = train_data[0][0].shape[1]

# Define CNN
class MultiClassImageClassifier(nn.Module):

    # Define the init method
    def __init__(self, num_classes):
        super(MultiClassImageClassifier, self).__init__()
        self.conv1 = nn.Conv2d(num_input_channels, num_output_channels, k
        self.relu = nn.ReLU()
```

```
        self.maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = nn.Flatten()

        # Create a fully connected layer
        self.fc = nn.Linear(num_output_channels * (image_size//2)**2, num

    def forward(self, x):
        # Pass inputs through each layer
        x = self.conv1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = self.flatten(x)
        x = self.fc(x)
        return x

# Define the training set DataLoader
dataloader_train = DataLoader(
    train_data,
    batch_size=10,
    shuffle=True,
)

# Define training function
def train_model(optimizer, net, num_epochs):
    num_processed = 0
    criterion = nn.CrossEntropyLoss()
    for epoch in range(num_epochs):
        running_loss = 0
        num_processed = 0
        for features, labels in dataloader_train:
            optimizer.zero_grad()
            outputs = net(features)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
            num_processed += len(labels)
        print(f'epoch {epoch}, loss: {running_loss / num_processed}')

    train_loss = running_loss / len(dataloader_train)


# Train for 1 epoch
net = MultiClassImageClassifier(num_classes)
optimizer = optim.Adam(net.parameters(), lr=0.001)

train_model(
    optimizer=optimizer,
    net=net,
    num_epochs=1,
)

# Test the model on the test set
```

```
# Define the test set DataLoader
dataloader_test = DataLoader(
    test_data,
    batch_size=10,
    shuffle=False,
)
# Define the metrics
accuracy_metric = Accuracy(task='multiclass', num_classes=num_classes)
precision_metric = Precision(task='multiclass', num_classes=num_classes,
recall_metric = Recall(task='multiclass', num_classes=num_classes, averag

# Run model on test set
net.eval()
predicted = []
for i, (features, labels) in enumerate(dataloader_test):
    output = net.forward(features.reshape(-1, 1, image_size, image_size))
    cat = torch.argmax(output, dim=-1)
    predicted.extend(cat.tolist())
    accuracy_metric(cat, labels)
    precision_metric(cat, labels)
    recall_metric(cat, labels)

# Compute the metrics
accuracy = accuracy_metric.compute().item()
precision = precision_metric.compute().tolist()
recall = recall_metric.compute().tolist()
print('Accuracy:', accuracy)
print('Precision (per class):', precision)
print('Recall (per class):', recall)
```

```
epoch 0, loss: 0.04051340300598337
Accuracy: 0.8773999810218811
Precision (per class): [0.7927509546279907, 0.99071204662323, 0.7688888907
432556, 0.802744448184967, 0.8444444537162781, 0.9768611788749695, 0.73374
2356300354, 0.9408283829689026, 0.9706774353981018, 0.9578736424446106]
Recall (per class): [0.8529999852180481, 0.9599999785423279, 0.86500000953
67432, 0.9359999895095825, 0.722000002861023, 0.9710000157356262, 0.597999
9899864197, 0.9539999961853027, 0.9599999785423279, 0.9549999833106995]
```