TOURE-Boubacar-Projet

February 18, 2023

1 Projet - Deep Learning

Auteur : TOURE Boubacar (boubacar.toure@etud.univ-angers.fr), étudiant en Master 2 Informatique à la faculté des sciences d'Angers

Professeur Référent : Sylvain Lamprier (sylvain.lamprier@univ-angers.fr)

2 Import du projet

[ ]:
```
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import torch.nn as nn
import torchvision
import zipfile
import tarfile
import shutil
import random
import torch
import time
import os

from tqdm import tqdm
from torchvision import models
from tqdm.autonotebook import tqdm
from torch.utils.tensorboard import SummaryWriter

from torchvision.models import ResNet18_Weights
from torchvision.models import AlexNet_Weights
from torchvision.models import SqueezeNet1_0_Weights
from torchvision.models import VGG16_Weights
from torchvision.models import DenseNet161_Weights
from torchvision.models import Inception_V3_Weights

# --->  A DECOMMENTER LES INSTRUCTIONS CI-DESSOUS SUR GOOGLE COLAB  <---
# Installez wget, Pillow, PIL, image avec pip
!pip install wget
!pip install Pillow==7.2.0
```

```
!pip install image
from PIL import Image

# Importation de la bibliothèque wget
import wget
```

3 Déclaration des fonctions

[16]:
```
def train(model, train_loader, validation_loader, loss_fn, optimizer, epochs=8,␣
 ↪typeTrain=""):
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    print("Device is :", device)

    model = model.to(device)
    loss_fn = loss_fn.to(device)

    # Clear any logs from previous runs
    if os.path.exists(f"/tmp/logs/project-train"):
        shutil.rmtree(f"/tmp/logs/project-train")

    # On créé un writer avec la date du modèle pour s'y retrouver
    TB_PATH = f"/tmp/logs/project"

    summary = SummaryWriter(f"{TB_PATH}-{typeTrain}-train")

    train_losses = []
    validation_losses = []
    accuraciesOfValidation = []
    accuraciesOfTrain = []

    start_time = time.time()
    print('\n', '-' * 40)
    for epoch in range(1, epochs+1):
        # Entraînement sur les données d'entraînement
        model.train()
        train_loss = 0.0
        correct = 0.0
        total = 0.0

        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            # Inception_v3 specific code
            if typeTrain == "inception_v3":
```

```
                outputs = outputs.logits  # extract the logits from␣
↪InceptionOutputs
            loss = loss_fn(outputs, labels)
            _, predicted = torch.max(outputs, 1)
            loss.backward()
            optimizer.step()
            train_loss += loss.item()
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        train_accuracy = correct / total
        accuraciesOfTrain.append(train_accuracy)
        summary.add_scalar("Accuracy_Of_Train", train_accuracy, epoch)
        summary.add_scalar("Loss_Of_Train", train_loss/len(train_loader), epoch)
        train_losses.append(train_loss/len(train_loader))

        if epoch % 2 == 0:
            # Evaluation sur les données de test
            model.eval()
            with torch.no_grad():
                validation_loss = 0.0
                correct = 0.0
                total = 0.0
                for images, labels in validation_loader:
                    images, labels = images.to(device), labels.to(device)
                    outputs = model(images)
                    validation_loss += loss_fn(outputs, labels).item()
                    _, predicted = torch.max(outputs, 1)
                    total += labels.size(0)
                    correct += (predicted == labels).sum().item()

                print('\t', '-' * 40)
                print("\tPredicted :", predicted)
                print("\tResult :", labels)
                print("\tTOTAL (correct/total):", correct, '/', total)
                print('\t', '-' * 40)

                validation_loss = validation_loss/len(validation_loader)
                validation_accuracy = correct / total
                accuraciesOfValidation.append(validation_accuracy)
                summary.add_scalar("Accuracy_Of_Validation",␣
↪validation_accuracy, epoch)
                summary.add_scalar("Loss_Of_Validation", validation_loss/
↪len(validation_loader), epoch)
                validation_losses.append(validation_loss/len(validation_loader))
```

```
                print("\tEpoch {}, Validation Loss: {:.4f}, Validation Accuracy:
↪ {:.4f} %".format(epoch, validation_loss, validation_accuracy * 100.0, '\n'))
        else:
            print("Epoch {}, Train Loss: {:.4f}, Train Accuracy: {:.4f} %".
↪format(epoch, train_loss/len(train_loader), train_accuracy * 100.0))
    print('\n', '-' * 40)
    end_time = time.time()

    trainTime = end_time - start_time
    trainModelComplexity = sum(p.numel() for p in model.parameters() if p.
↪requires_grad)

    plt.figure(figsize=(20, 8))
    plt.subplot(121)
    plt.plot(train_losses)
    plt.title('Train Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid()

    plt.subplot(122)
    plt.plot(accuraciesOfTrain)
    plt.title('Train Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid()
    plt.show()

    plt.figure(figsize=(20, 8))
    plt.subplot(121)
    plt.plot(validation_losses)
    plt.title('Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid()

    plt.subplot(122)
    plt.plot(accuraciesOfValidation)
    plt.title('Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid()
    plt.show()

    # Load the TensorBoard notebook extension
    %load_ext tensorboard
    %tensorboard --logdir {TB_PATH}-{typeTrain}-train
```

```
    return trainTime, trainModelComplexity, validation_loss,␣
 ↪validation_accuracy, accuraciesOfValidation

def prepare_data(mean, std, size_of_image, transform, batch_size, split):
    # Chargement des données dans le dossier 101_ObjectCategories
    dataset = torchvision.datasets.ImageFolder(root='./101_ObjectCategories',␣
 ↪transform=transform)

    # Chargement des données dans le dossier 101_ObjectCategories
    dataset = torchvision.datasets.ImageFolder(root='./101_ObjectCategories',␣
 ↪transform=transform)

    # Usage de Subset pour la formation des données d'entrainement et de tests␣
 ↪de validations
    indices = list(range(len(dataset)))
    random.shuffle(indices)
    split = int(split * len(indices))
    train_indices = indices[:split]
    test_indices = indices[split:]
    train_dataset = torch.utils.data.Subset(dataset, train_indices)
    test_dataset = torch.utils.data.Subset(dataset, test_indices)

    # Création des dataloaders pour les données d'entrainement et de tests de␣
 ↪validations (via torch)
    train_dataloader = torch.utils.data.DataLoader(train_dataset,␣
 ↪batch_size=batch_size, shuffle=True, pin_memory=True, drop_last=True)
    test_dataloader = torch.utils.data.DataLoader(test_dataset,␣
 ↪batch_size=batch_size, shuffle=True, pin_memory=True, drop_last=True)

    return train_dataloader, test_dataloader

def prepare_data_on_google_colab(mean, std, size_of_image, transform,␣
 ↪batch_size, split):
    # Téléchargez l'archive
    if not os.path.exists("caltech-101.zip"):
        url = "https://data.caltech.edu/records/mzrjq-6wc02/files/caltech-101.
 ↪zip"
        wget.download(url)

    # Retirez le répertoire "101_ObjectCategories"
    if os.path.exists("101_ObjectCategories"):
        shutil.rmtree("101_ObjectCategories")

    with zipfile.ZipFile('caltech-101.zip', 'r') as zip_ref:
        zip_ref.extractall()
```

```
    with tarfile.open('caltech-101/101_ObjectCategories.tar.gz', "r:gz") as tar:
        tar.extractall()

    # Retirez le répertoire "caltech-101"
    if os.path.exists("caltech-101"):
        shutil.rmtree("caltech-101")

    # Retirez le répertoire "BACKGROUND_Google"
    if os.path.exists("101_ObjectCategories/BACKGROUND_Google"):
        shutil.rmtree("101_ObjectCategories/BACKGROUND_Google")

    print("La liste des fichiers du dossier '/content/' sur colab :", os.
↪listdir())

    # Chargement des données dans le dossier 101_ObjectCategories
    dataset = torchvision.datasets.ImageFolder(root='./101_ObjectCategories',␣
↪transform=transform)

    # Usage de Subset pour la formation des données d'entrainement et de tests␣
↪de validations
    indices = list(range(len(dataset)))
    random.shuffle(indices)
    split = int(split * len(indices))
    train_indices = indices[:split]
    test_indices = indices[split:]
    train_dataset = torch.utils.data.Subset(dataset, train_indices)
    test_dataset = torch.utils.data.Subset(dataset, test_indices)

    # Création des dataloaders pour les données d'entrainement et de tests de␣
↪validations (via torch)
    train_dataloader = torch.utils.data.DataLoader(train_dataset,␣
↪batch_size=batch_size, shuffle=True, pin_memory=True, drop_last=True)
    test_dataloader = torch.utils.data.DataLoader(test_dataset,␣
↪batch_size=batch_size, shuffle=True, pin_memory=True, drop_last=True)

    return train_dataloader, test_dataloader

def adapt_pretrained_model(model_name, num_classes=101):
    # Load the pretrained model
    if model_name == "resnet18":
        model = models.resnet18(weights=ResNet18_Weights.IMAGENET1K_V1)
        model.fc = torch.nn.Linear(model.fc.in_features, num_classes)
    elif model_name == "alexnet":
        model = models.alexnet(weights=AlexNet_Weights.IMAGENET1K_V1)
```

```
        model.classifier[-1] = torch.nn.Linear(model.classifier[-1].
↪in_features, num_classes)
    elif model_name == "squeezenet1_0":
        model = models.squeezenet1_0(weights=SqueezeNet1_0_Weights.
↪IMAGENET1K_V1)
        model.classifier[1] = torch.nn.Conv2d(512, num_classes, kernel_size=(1,␣
↪1), stride=(1, 1))
        model.num_classes = num_classes
    elif model_name == "vgg16":
        model = models.vgg16(weights=VGG16_Weights.IMAGENET1K_V1)
        model.classifier[-1] = torch.nn.Linear(model.classifier[-1].
↪in_features, num_classes)
    elif model_name == "densenet161":
        model = models.densenet161(weights=DenseNet161_Weights.IMAGENET1K_V1)
        model.classifier = torch.nn.Linear(in_features=2208,␣
↪out_features=num_classes, bias=True)
        model.num_classes = num_classes
    elif model_name == "inception_v3":
        model = models.inception_v3(weights=Inception_V3_Weights.IMAGENET1K_V1)
        model.fc = torch.nn.Linear(model.fc.in_features, num_classes)
    else:
        raise ValueError("Model not recognized")

    for name, param in model.named_parameters():
      if name.startswith('layer1'):
          param.requires_grad = False

    return model

def compare_models(executionTimeOfModel1, complexityOfModel1,␣
↪validationLossOfModel1, accuracyOfModel1, executionTimeOfModel2,␣
↪complexityOfModel2, validationLossOfModel2, accuracyOfModel2):
    print("Le modèle 1 a une perte de {} sur les données de validation avec une␣
↪précision de {}% et une complexité de {} paramètres et un temps d'évaluation␣
↪de {}s".format(validationLossOfModel1, accuracyOfModel1*100.0,␣
↪complexityOfModel1, executionTimeOfModel1))
    print("\nLe modèle 2 a une perte de {} sur les données de validation avec␣
↪une précision de {}% et une complexité de {} paramètres et un temps␣
↪d'évaluation de {}s".format(validationLossOfModel2, accuracyOfModel2*100.0,␣
↪complexityOfModel2, executionTimeOfModel2))

    if validationLossOfModel1 < validationLossOfModel2:
        print("\n\tLe modèle 1 a une perte plus faible sur les données de␣
↪validation.")
        return 1
    else:
```

```
        print("\n\tLe modèle 2 a une perte plus faible sur les données de␣
 ↪validation.")
        return 2

def affiche_dataloader(dataloader):
    for i, data in enumerate(dataloader):
        if i <= 10:
            images, labels = data
            print("Image Shape: ", images.shape)
            print("Label Shape: ", labels.shape)
        else:
            break
```

4 Chargement des données du projet

[18]:
```
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
size_of_image = [224, 224]

batch_size = 16
split = 0.9
epochs = 8

transform = transforms.Compose([
    transforms.Resize(size_of_image),
    transforms.CenterCrop(size_of_image),
    transforms.ToTensor(),
    transforms.Normalize(mean, std)
])

# train_dataloader, test_dataloader = prepare_data(mean, std, size_of_image,␣
 ↪transform, batch_size, split)
train_dataloader, test_dataloader = prepare_data_on_google_colab(mean, std,␣
 ↪size_of_image, transform, batch_size, split)
```

```
La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
```

5 Entrainement et chargement des modèles

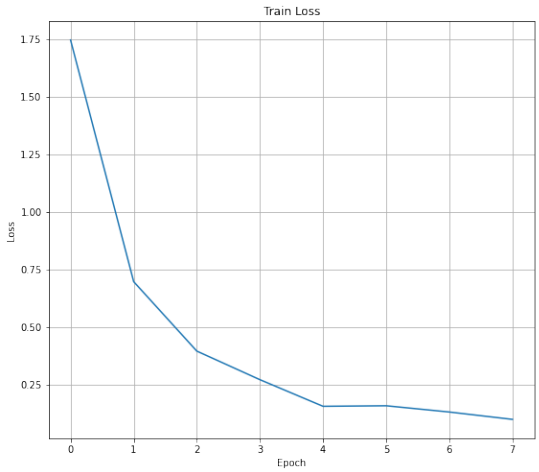
5.1 Le modèle « resnet18 » avec l'optimiseur : Adam

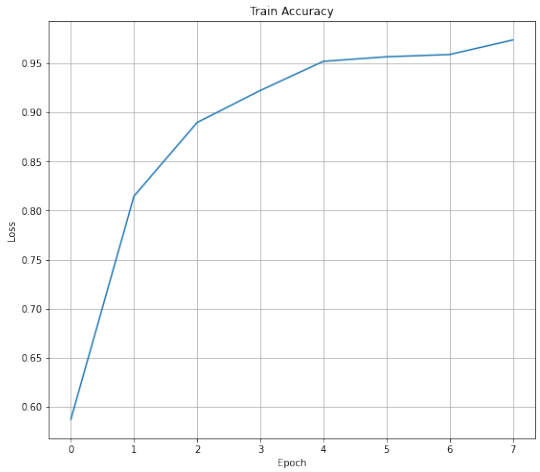
[ ]:
```
modelAdam_resnet18 = adapt_pretrained_model("resnet18")
loss_fn = nn.CrossEntropyLoss()
optimiseurAdam_resnet18 = torch.optim.Adam(modelAdam_resnet18.parameters(),␣
 ↪lr=0.001)
```

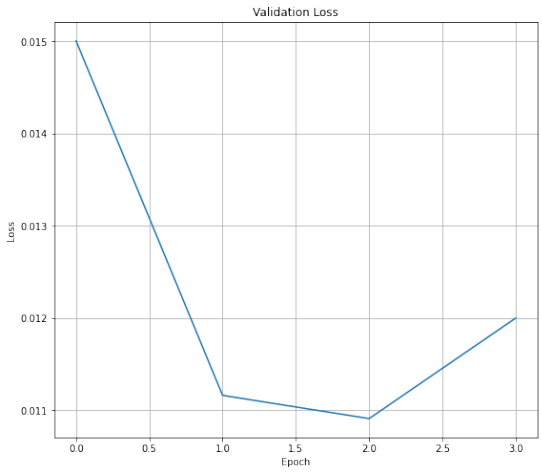
```
modelAdam_resnet18_trainTime, modelAdam_resnet18_trainModelComplexity,␣
↪modelAdam_resnet18_validation_loss, modelAdam_resnet18_accuracy, _ =␣
↪train(modelAdam_resnet18, train_dataloader, test_dataloader, loss_fn,␣
↪optimiseurAdam_resnet18, epochs, typeTrain="resnet18-Adam")
```

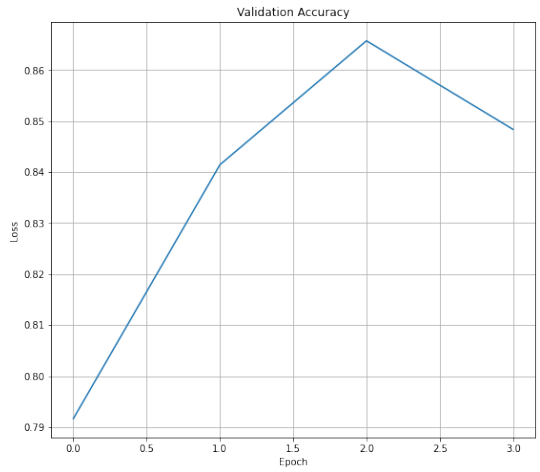
```
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to
/root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth

  0%|          | 0.00/44.7M [00:00<?, ?B/s]

Device is : cuda:0


 ----------------------------------------
Epoch 1, Train Loss: 1.7480, Train Accuracy: 58.7090 %
         ----------------------------------------
        Predicted : tensor([ 0, 29, 43, 59, 48, 16,  3,  1, 75,  3, 99, 72, 11,
66, 19,  0],
       device='cuda:0')
        Result : tensor([ 0, 28, 43, 61, 14, 66,  3,  1, 75,  3, 85, 72, 56, 66,
19,  0],
       device='cuda:0')
        TOTAL (correct/total): 684.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.8104, Validation Accuracy: 79.1667 %
Epoch 3, Train Loss: 0.3952, Train Accuracy: 88.9728 %
         ----------------------------------------
        Predicted : tensor([72,  0, 25,  5, 81, 25, 14,  3, 36,  3, 22,  0,  7,
13, 85,  1],
       device='cuda:0')
        Result : tensor([25,  0, 25,  5, 81, 95, 14,  3, 14,  3, 22,  0, 23, 93,
85,  1],
       device='cuda:0')
        TOTAL (correct/total): 727.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.6028, Validation Accuracy: 84.1435 %
Epoch 5, Train Loss: 0.1551, Train Accuracy: 95.2100 %
         ----------------------------------------
        Predicted : tensor([87,  3, 94, 15,  5, 27,  0, 29, 26, 16, 30,  3,  5,
2, 59, 43],
       device='cuda:0')
        Result : tensor([87,  3, 94, 15,  5, 81,  0, 29, 26, 16, 11,  3,  5,  2,
59, 43],
       device='cuda:0')
        TOTAL (correct/total): 748.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.5892, Validation Accuracy: 86.5741 %
Epoch 7, Train Loss: 0.1303, Train Accuracy: 95.9016 %
         ----------------------------------------
        Predicted : tensor([16, 26,  3, 71, 31, 94, 32, 34, 68,  1, 77, 56,  5,
```

```
26,  5,  1],
       device='cuda:0')
        Result : tensor([16, 26,  3, 71, 31, 94, 32, 34, 68,  1, 77, 56,  5, 77,
5,  1],
       device='cuda:0')
        TOTAL (correct/total): 733.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.6480, Validation Accuracy: 84.8380 %

 ----------------------------------------
```









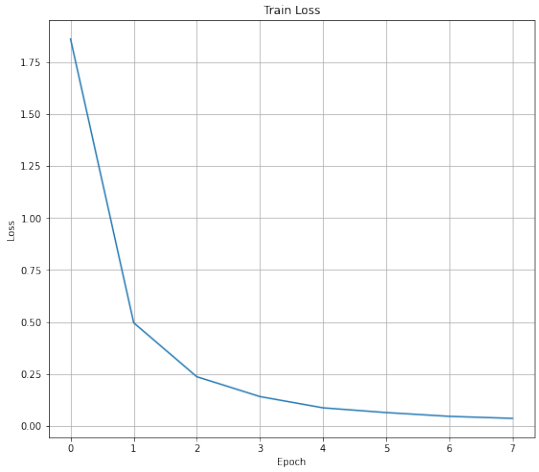

```
<IPython.core.display.Javascript object>
```

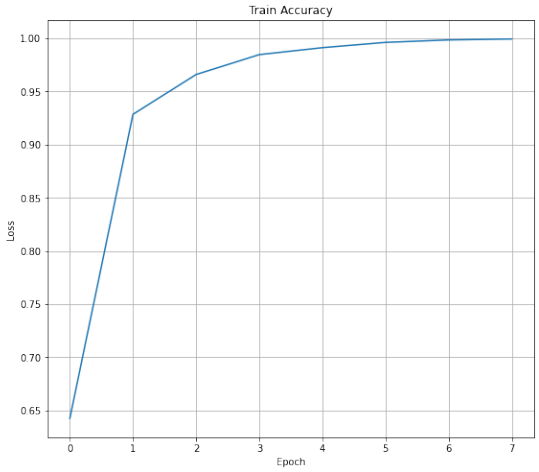
5.2 Le modèle « resnet18 » avec l'optimiseur : SGD

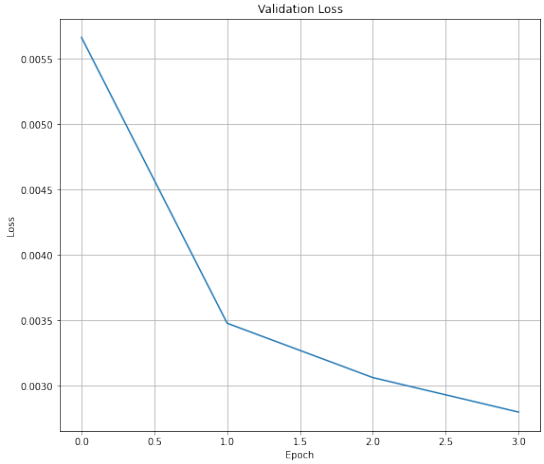
[ ]:
```
model_resnet18 = adapt_pretrained_model("resnet18")
loss_fn = nn.CrossEntropyLoss()
optimiseur_resnet18 = torch.optim.SGD(model_resnet18.parameters(), lr=0.001,␣
 ↪momentum=0.9)
model_resnet18_trainTime, model_resnet18_trainModelComplexity,␣
 ↪model_resnet18_validation_loss, model_resnet18_accuracy, _ =␣
 ↪train(model_resnet18, train_dataloader, test_dataloader, loss_fn,␣
 ↪optimiseur_resnet18, epochs, typeTrain="resnet18")
```

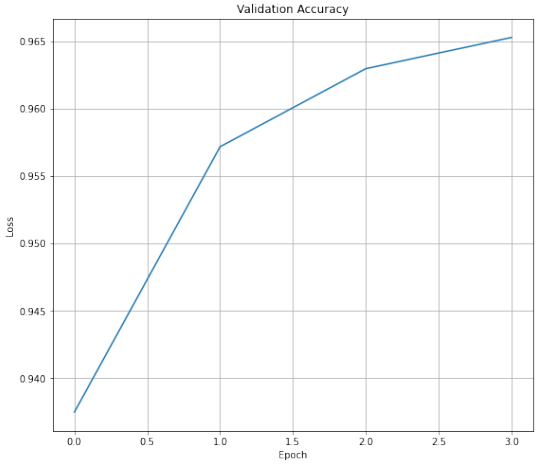
```
Device is : cuda:0

 ---------------------------------------
Epoch 1, Train Loss: 1.8621, Train Accuracy: 64.2674 %
         ---------------------------------------
        Predicted : tensor([28, 88,  5,  1, 77,  0,  0, 56,  1, 98,  4,  2,  5,
54, 94, 82],
       device='cuda:0')
        Result : tensor([28, 88,  5,  1, 77,  0,  0, 56,  1, 98,  4,  2,  5, 54,
94, 67],
       device='cuda:0')
        TOTAL (correct/total): 810.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.3059, Validation Accuracy: 93.7500 %
Epoch 3, Train Loss: 0.2362, Train Accuracy: 96.5932 %
         ---------------------------------------
        Predicted : tensor([ 5, 27, 13, 66, 77, 59,  8, 97, 94,  5,  5,  5,  5,
27, 69,  5],
       device='cuda:0')
        Result : tensor([ 5, 27, 43, 66, 77, 59,  8, 97, 94,  5,  5,  5,  5, 27,
69,  5],
       device='cuda:0')
        TOTAL (correct/total): 827.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.1879, Validation Accuracy: 95.7176 %
Epoch 5, Train Loss: 0.0864, Train Accuracy: 99.1163 %
         ---------------------------------------
        Predicted : tensor([72, 88,  1, 57,  0, 53, 93, 13,  5,  2,  0,  1, 12,
79,  3, 23],
       device='cuda:0')
        Result : tensor([72, 88,  1, 57,  0, 53, 56, 13,  5,  2,  0,  1, 12, 79,
3, 23],
       device='cuda:0')
        TOTAL (correct/total): 832.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.1655, Validation Accuracy: 96.2963 %
Epoch 7, Train Loss: 0.0455, Train Accuracy: 99.8463 %
```

```
        ----------------------------------------
       Predicted : tensor([ 12,  30,   3,  77,  72,  71,  51,  43,  47,   5,
33, 100,  98,   1,
        86,  28], device='cuda:0')
       Result : tensor([12, 30,  3, 77, 72, 71, 51, 43, 47,  5, 10, 66, 98,  1,
86, 28],
      device='cuda:0')
       TOTAL (correct/total): 834.0 / 864.0
        ----------------------------------------
       Epoch 8, Validation Loss: 0.1512, Validation Accuracy: 96.5278 %

----------------------------------------
```










```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
<IPython.core.display.Javascript object>
```

5.3 Comparaison : resnet18 Adam VS resnet18 SGD

[ ]:
```
resultTest1 = compare_models(modelAdam_resnet18_trainTime,␣
 ↪modelAdam_resnet18_trainModelComplexity, modelAdam_resnet18_validation_loss,␣
 ↪modelAdam_resnet18_accuracy, model_resnet18_trainTime,␣
 ↪model_resnet18_trainModelComplexity, model_resnet18_validation_loss,␣
 ↪model_resnet18_accuracy)

if resultTest1 == 1:
    print("\n_____\nLe Modèle␣
 ↪resnet18 Adam est le plus performant !
 ↪\n_____\n")
else:
    print("\n_____\nLe Modèle␣
 ↪resnet18 SGD est le plus performant !
 ↪\n_____\n")
```

```
Le modèle 1 a une perte de 0.6480460575333348 sur les données de validation avec
une précision de 84.83796296296296% et une complexité de 11080357 paramètres et
un temps d'évaluation de 423.7692291736603s

Le modèle 2 a une perte de 0.15122037219245815 sur les données de validation
avec une précision de 96.52777777777779% et une complexité de 11080357
paramètres et un temps d'évaluation de 380.01206827163696s

        Le modèle 2 a une perte plus faible sur les données de validation.


-----------------------------------------------------
Le Modèle resnet18 SGD est le plus performant !
-----------------------------------------------------
```

5.4 Conclusion :

A la suite de nombreux tests que nous avons eu à réaliser en utilisant le separateur **"torch.utils.data.Subset"**. Nous avons remarquer que l'usage de l'optimiseur SGD est le plus adapté pour la réalisation de nos tests. Voilà pourquoi, tous les tests que nous allons faire à partir de maintenant se feront avec **l'optimiseur SGD en priorité**.

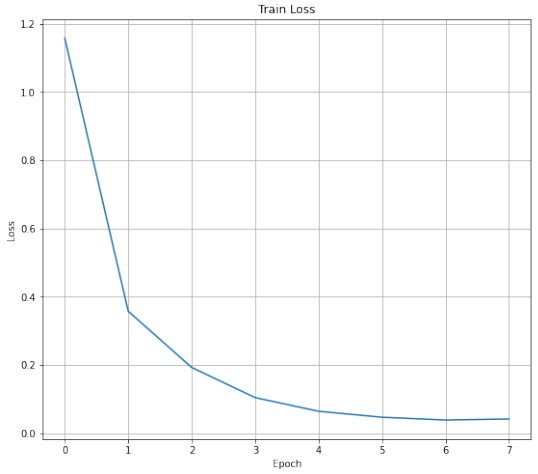
5.5 Le modèle « alexnet » avec l'optimiseur : SGD

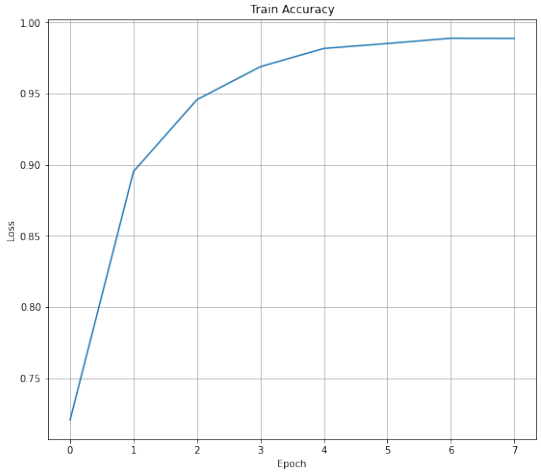
[ ]:
```
model_alexnet = adapt_pretrained_model("alexnet")
loss_fn = nn.CrossEntropyLoss()
optimiseur_alexnet = torch.optim.SGD(model_alexnet.parameters(), lr=0.001,␣
 ↪momentum=0.9)
```

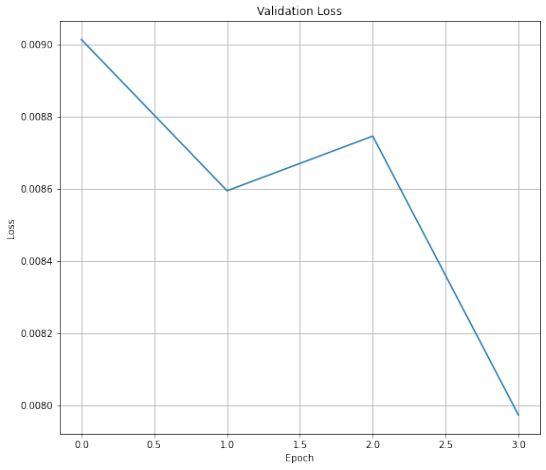
```
model_alexnet_trainTime, model_alexnet_trainModelComplexity,␣
 ↪model_alexnet_validation_loss, model_alexnet_accuracy, _ =␣
 ↪train(model_alexnet, train_dataloader, test_dataloader, loss_fn,␣
 ↪optimiseur_alexnet, epochs, typeTrain="alexnet")
```

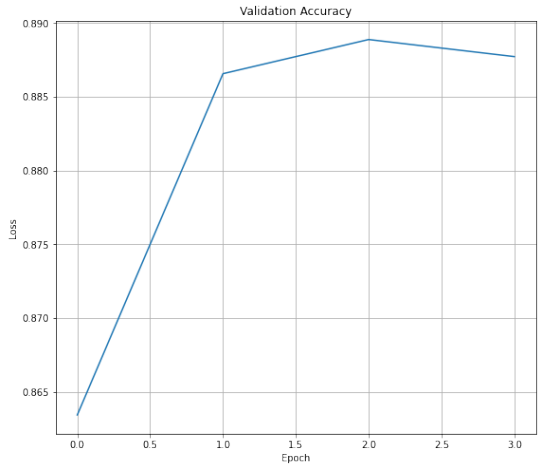
```
Downloading: "https://download.pytorch.org/models/alexnet-owt-7be5be79.pth" to
/root/.cache/torch/hub/checkpoints/alexnet-owt-7be5be79.pth

  0%|          | 0.00/233M [00:00<?, ?B/s]

Device is : cuda:0


 ----------------------------------------
Epoch 1, Train Loss: 1.1574, Train Accuracy: 72.0671 %
         ----------------------------------------
        Predicted : tensor([  3,  24,  98,  88, 100,   2,   3,  76,  23,  94,
18,  44,   2,   3,
          5,  85], device='cuda:0')
        Result : tensor([  3,  24,  98,  88, 100,   2,   3,  76,  23,  38,  18,
44,   2,   3,
          5,  35], device='cuda:0')
        TOTAL (correct/total): 746.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.4867, Validation Accuracy: 86.3426 %
Epoch 3, Train Loss: 0.1931, Train Accuracy: 94.5825 %
         ----------------------------------------
        Predicted : tensor([13, 66,  3, 82, 48, 43, 44, 32,  5, 38, 26, 77, 63,
5,  5, 33],
       device='cuda:0')
        Result : tensor([13, 66,  3, 82, 48, 43, 44, 32,  5, 38, 26, 77, 63,  5,
5, 33],
       device='cuda:0')
        TOTAL (correct/total): 766.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.4641, Validation Accuracy: 88.6574 %
Epoch 5, Train Loss: 0.0648, Train Accuracy: 98.1814 %
         ----------------------------------------
        Predicted : tensor([72, 39, 11, 24, 56,  3, 50,  0, 89, 87, 94, 92, 98,
5,  5,  2],
       device='cuda:0')
        Result : tensor([72, 39, 11, 10, 56,  3, 23,  0, 89, 87, 94, 92, 98,  5,
5,  2],
       device='cuda:0')
        TOTAL (correct/total): 768.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.4723, Validation Accuracy: 88.8889 %
Epoch 7, Train Loss: 0.0392, Train Accuracy: 98.8986 %
         ----------------------------------------
        Predicted : tensor([ 2,  3, 40,  0,  0,  3, 31, 50, 21,  3, 49, 83, 86,
```

```
9, 74,  0],
       device='cuda:0')
        Result : tensor([ 2,  3, 40,  0,  0,  3, 31, 50, 85,  3, 49, 17, 86, 97,
74,  0],
       device='cuda:0')
        TOTAL (correct/total): 767.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.4305, Validation Accuracy: 88.7731 %

 ----------------------------------------
```










```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

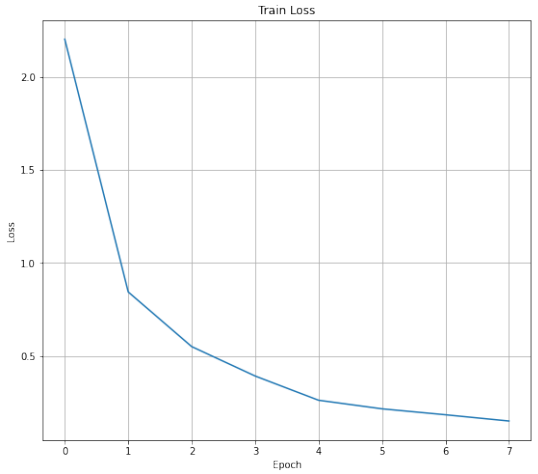
```
<IPython.core.display.Javascript object>
```

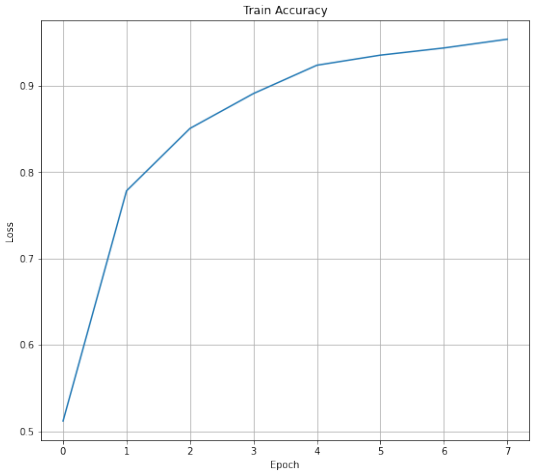
5.6 Le modèle « squeezenet1_0 » avec l'optimiseur : SGD

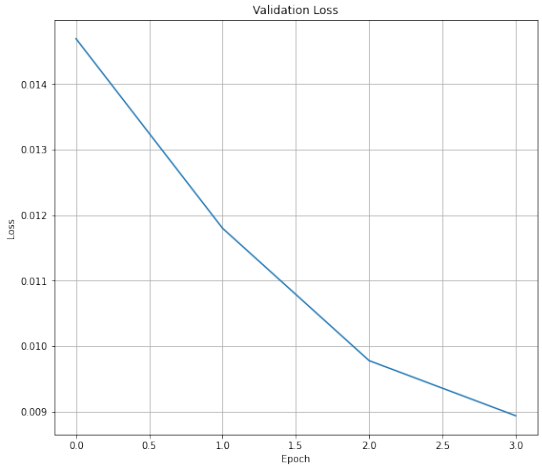
[ ]:
```
model_squeezenet1_0 = adapt_pretrained_model("squeezenet1_0")
loss_fn = nn.CrossEntropyLoss()
optimiseur_squeezenet1_0 = torch.optim.SGD(model_squeezenet1_0.parameters(),␣
 ↪lr=0.001, momentum=0.9)
model_squeezenet1_0_trainTime, model_squeezenet1_0_trainModelComplexity,␣
 ↪model_squeezenet1_0_validation_loss, model_squeezenet1_0_accuracy, _ =␣
 ↪train(model_squeezenet1_0, train_dataloader, test_dataloader, loss_fn,␣
 ↪optimiseur_squeezenet1_0, epochs, typeTrain="squeezenet1_0")
```

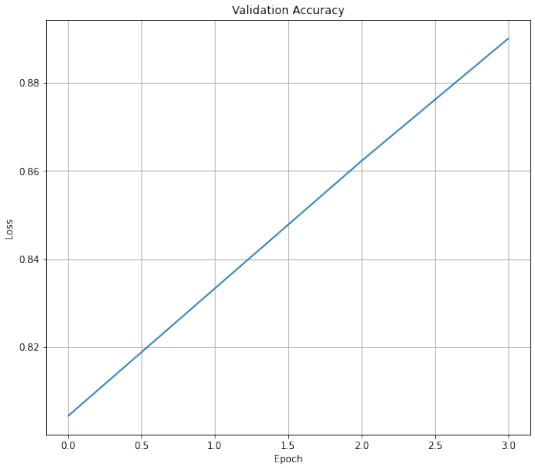
```
Downloading: "https://download.pytorch.org/models/squeezenet1_0-b66bff10.pth" to
/root/.cache/torch/hub/checkpoints/squeezenet1_0-b66bff10.pth

  0%|          | 0.00/4.78M [00:00<?, ?B/s]

Device is : cuda:0


 ----------------------------------------
Epoch 1, Train Loss: 2.2011, Train Accuracy: 51.1911 %
         ----------------------------------------
        Predicted : tensor([77,  0, 71,  1, 35, 97, 72, 35, 97,  3, 53,  0,  3,
94,  3, 71],
       device='cuda:0')
        Result : tensor([77,  0, 71,  1, 35, 97, 89, 35, 97,  3, 53,  0,  3, 94,
3, 71],
       device='cuda:0')
        TOTAL (correct/total): 695.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.7936, Validation Accuracy: 80.4398 %
Epoch 3, Train Loss: 0.5518, Train Accuracy: 85.0538 %
         ----------------------------------------
        Predicted : tensor([11, 41, 47, 40,  1, 94, 56, 35, 89,  0,  3, 46, 94,
35, 36,  3],
       device='cuda:0')
        Result : tensor([11, 41, 47, 40,  1, 94, 56, 35, 89,  0,  3, 46, 94, 61,
36,  3],
       device='cuda:0')
        TOTAL (correct/total): 720.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.6371, Validation Accuracy: 83.3333 %
Epoch 5, Train Loss: 0.2638, Train Accuracy: 92.3668 %
         ----------------------------------------
        Predicted : tensor([  2,   2,   2,  98,  39,  48,  82,  29,  43,  61,
88,   1,   3,   3,
          3, 100], device='cuda:0')
        Result : tensor([  2,   2,   2,  98,  39,  48,  82,  29,  43,   5,  88,
1,   3,   3,
          3, 100], device='cuda:0')
```

```
        TOTAL (correct/total): 745.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.5279, Validation Accuracy: 86.2269 %
Epoch 7, Train Loss: 0.1860, Train Accuracy: 94.3648 %
         ---------------------------------------
        Predicted : tensor([87, 74,  6, 11,  5, 28, 93, 43, 94,  3,  3,  0, 63,
27,  5, 16],
       device='cuda:0')
        Result : tensor([87, 74,  6, 11,  5, 47, 73, 43, 94,  3,  3,  0, 63, 27,
5, 16],
       device='cuda:0')
        TOTAL (correct/total): 769.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.4825, Validation Accuracy: 89.0046 %

 ----------------------------------------
```









```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

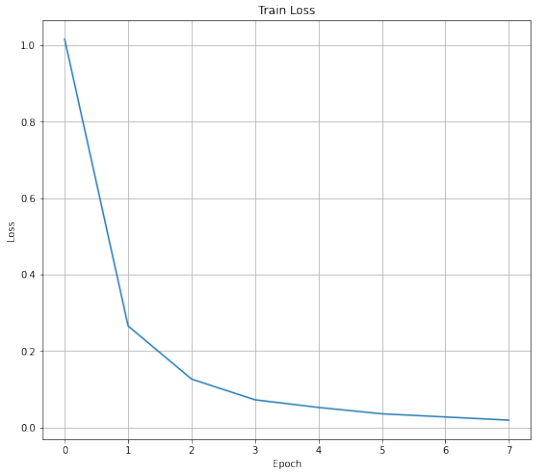
```
<IPython.core.display.Javascript object>
```

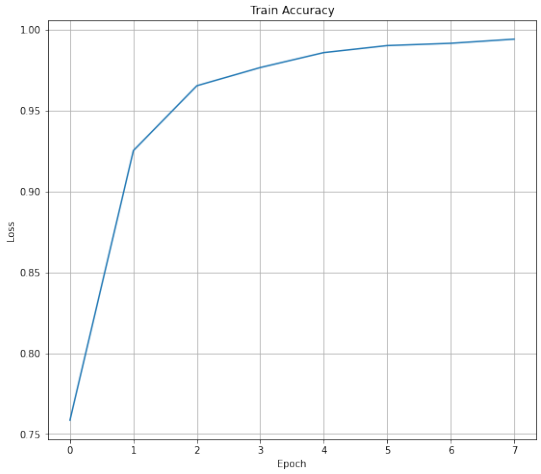
5.7 Le modèle « vgg16 » avec l'optimiseur : SGD

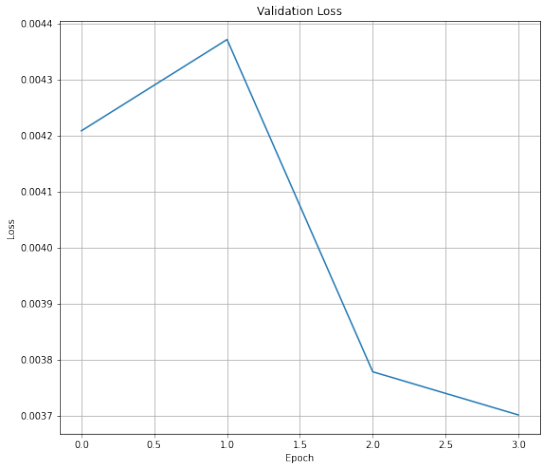
[ ]:
```
model_vgg16 = adapt_pretrained_model("vgg16")
loss_fn = nn.CrossEntropyLoss()
optimiseur_vgg16 = torch.optim.SGD(model_vgg16.parameters(), lr=0.001,␣
 ↪momentum=0.9)
model_vgg16_trainTime, model_vgg16_trainModelComplexity,␣
 ↪model_vgg16_validation_loss, model_vgg16_accuracy, _ = train(model_vgg16,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_vgg16, epochs,␣
 ↪typeTrain="vgg16")
```

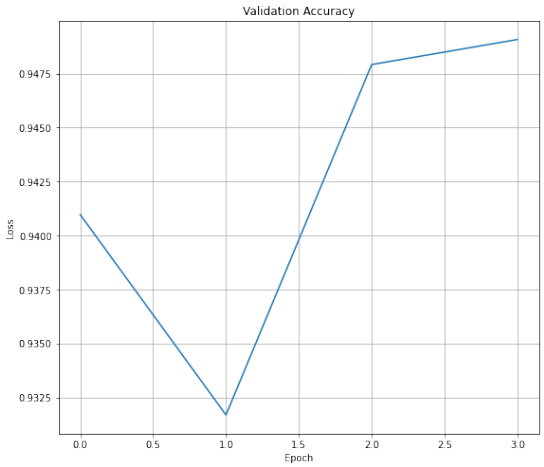
```
Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to
/root/.cache/torch/hub/checkpoints/vgg16-397923af.pth

  0%|          | 0.00/528M [00:00<?, ?B/s]

Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 1.0160, Train Accuracy: 75.8581 %
         ----------------------------------------
        Predicted : tensor([81, 50, 60,  5,  5, 79, 48, 66, 31, 47, 27, 57,  2,
3, 63,  3],
       device='cuda:0')
        Result : tensor([81, 50, 60,  5,  5, 79, 48, 66, 31, 47, 28, 57,  2,  3,
63,  3],
       device='cuda:0')
        TOTAL (correct/total): 813.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.2273, Validation Accuracy: 94.0972 %
Epoch 3, Train Loss: 0.1268, Train Accuracy: 96.5292 %
         ----------------------------------------
        Predicted : tensor([11, 98,  1, 47, 36, 54, 50, 34, 47, 88,  1, 26, 46,
94,  1, 39],
       device='cuda:0')
        Result : tensor([11, 98,  1, 47, 36, 54, 50, 34, 47, 88,  1, 81, 46, 94,
1, 39],
       device='cuda:0')
        TOTAL (correct/total): 805.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.2361, Validation Accuracy: 93.1713 %
Epoch 5, Train Loss: 0.0526, Train Accuracy: 98.5784 %
         ----------------------------------------
```

```
        Predicted : tensor([98,  1,  2, 58,  3, 39,  0, 47, 55, 52, 41, 82, 66,
60,  0,  5],
       device='cuda:0')
        Result : tensor([98,  1,  2, 58,  3, 39,  0, 47, 55, 52, 41, 67, 66, 60,
0,  5],
       device='cuda:0')
        TOTAL (correct/total): 819.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.2040, Validation Accuracy: 94.7917 %
Epoch 7, Train Loss: 0.0277, Train Accuracy: 99.1547 %
         ----------------------------------------
        Predicted : tensor([ 0, 35,  5, 22,  3, 66, 41, 64,  5, 12, 25, 31, 27,
1,  6,  2],
       device='cuda:0')
        Result : tensor([ 0, 35,  5, 22,  3, 66, 41, 64,  5, 12, 25, 31, 81,  1,
6,  2],
       device='cuda:0')
        TOTAL (correct/total): 820.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.1999, Validation Accuracy: 94.9074 %

 ----------------------------------------
```










```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

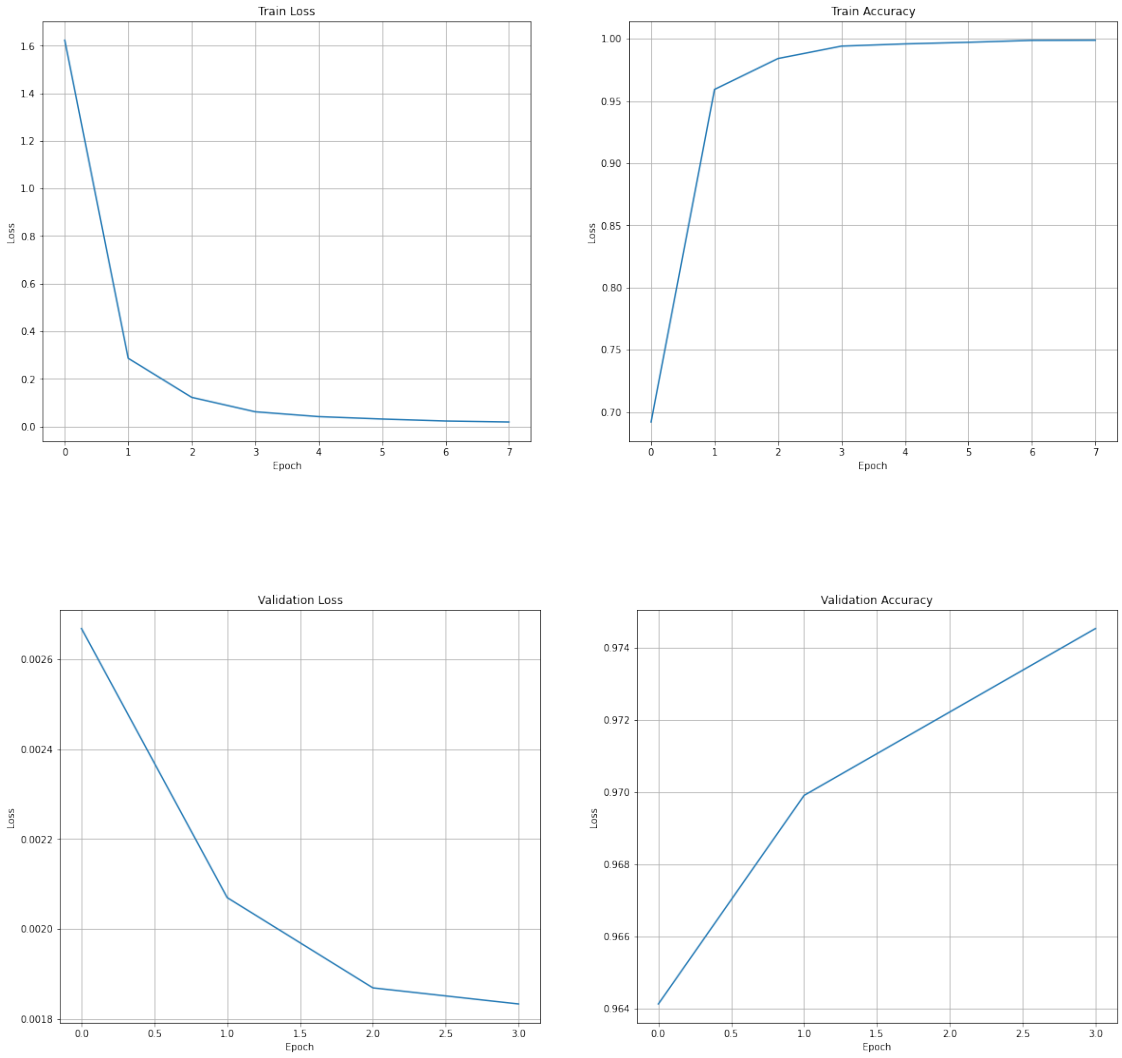
```
<IPython.core.display.Javascript object>
```

5.8 Le modèle « densenet161 » avec l'optimiseur : SGD

[ ]:
```
model_densenet161 = adapt_pretrained_model("densenet161")
loss_fn = nn.CrossEntropyLoss()
optimiseur_densenet161 = torch.optim.SGD(model_densenet161.parameters(), lr=0.
 ↪001, momentum=0.9)
model_densenet161_trainTime, model_densenet161_trainModelComplexity,␣
 ↪model_densenet161_validation_loss, model_densenet161_accuracy, _ =␣
 ↪train(model_densenet161, train_dataloader, test_dataloader, loss_fn,␣
 ↪optimiseur_densenet161, epochs, typeTrain="densenet161")
```

```
Downloading: "https://download.pytorch.org/models/densenet161-8d451a50.pth" to
/root/.cache/torch/hub/checkpoints/densenet161-8d451a50.pth

  0%|          | 0.00/110M [00:00<?, ?B/s]

Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 1.6230, Train Accuracy: 69.1855 %
         ----------------------------------------
        Predicted : tensor([77, 88,  0, 72, 81, 72,  1, 41,  1, 22, 44,  5, 85,
74, 65, 57],
       device='cuda:0')
        Result : tensor([77, 88,  0, 89, 81, 72,  1, 41,  1, 22, 44,  5, 85, 74,
65, 57],
       device='cuda:0')
        TOTAL (correct/total): 833.0 / 864.0
```

```
 ---------------------------------------
 Epoch 2, Validation Loss: 0.1441, Validation Accuracy: 96.4120 %
Epoch 3, Train Loss: 0.1220, Train Accuracy: 98.4119 %
 ---------------------------------------
 Predicted : tensor([51, 78,  5, 18, 43,  3, 34, 49, 55, 23,  5,  1, 23,
44, 26,  5],
       device='cuda:0')
 Result : tensor([51, 78,  5, 18, 43,  3, 34, 49, 79, 23,  5,  1, 23, 44,
26,  5],
       device='cuda:0')
 TOTAL (correct/total): 838.0 / 864.0
 ---------------------------------------
 Epoch 4, Validation Loss: 0.1118, Validation Accuracy: 96.9907 %
Epoch 5, Train Loss: 0.0411, Train Accuracy: 99.5902 %
 ---------------------------------------
 Predicted : tensor([32,  3, 28,  3, 56,  5, 68, 77, 81,  2, 67, 31,  2,
0, 55, 28],
       device='cuda:0')
 Result : tensor([32,  3, 28,  3, 56,  5, 68, 77, 81,  2, 67, 31,  2,  0,
55, 28],
       device='cuda:0')
 TOTAL (correct/total): 840.0 / 864.0
 ---------------------------------------
 Epoch 6, Validation Loss: 0.1009, Validation Accuracy: 97.2222 %
Epoch 7, Train Loss: 0.0226, Train Accuracy: 99.8719 %
 ---------------------------------------
 Predicted : tensor([65, 54, 27, 56,  3, 90, 26, 25, 72, 44, 81, 79, 26,
98, 13, 95],
       device='cuda:0')
 Result : tensor([65, 54, 27, 56,  3, 90, 26, 25, 89, 44, 81, 79, 26, 98,
13, 95],
       device='cuda:0')
 TOTAL (correct/total): 842.0 / 864.0
 ---------------------------------------
 Epoch 8, Validation Loss: 0.0990, Validation Accuracy: 97.4537 %

 ---------------------------------------
```




```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
<IPython.core.display.Javascript object>
```

5.9 Le modèle « inception_v3 » avec l'optimiseur : SGD

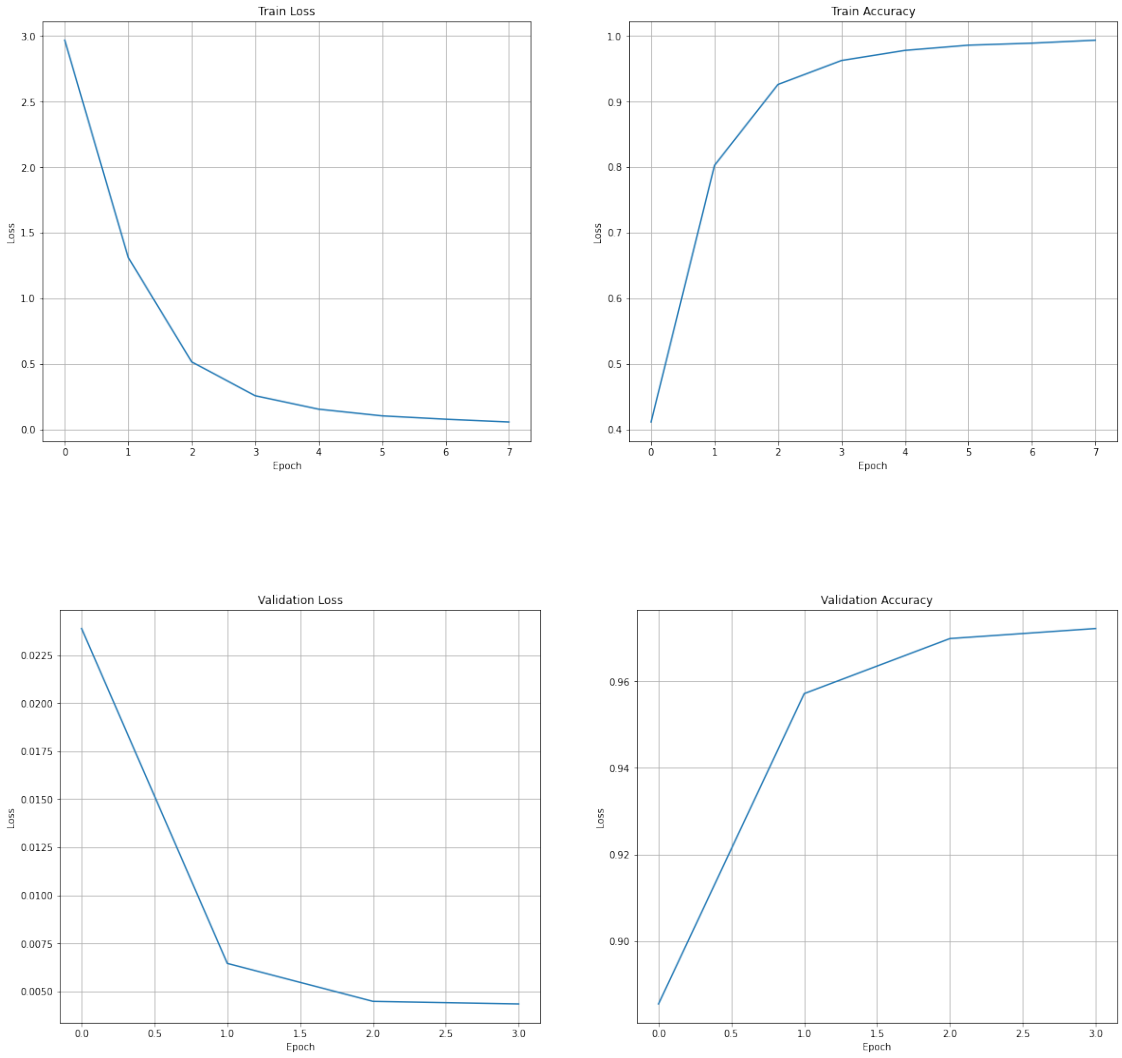
[11]:
```
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
size_of_image = [299, 299]

batch_size = 32
split = 0.9
epochs = 8
```

```
transform = transforms.Compose([
    transforms.Resize(size_of_image),
    transforms.ToTensor(),
    transforms.Normalize(mean, std)
])

# affiche_dataloader(test_dataloader)

train_dataloader_v3, test_dataloader_v3 = prepare_data(mean, std,␣
 ↪size_of_image, transform, batch_size, split)
# train_dataloader_v3, test_dataloader_v3 =␣
 ↪prepare_data_on_google_colab(mean=mean, std=std,␣
 ↪size_of_image=size_of_image, transform=transform, batch_size=batch_size,␣
 ↪split=split)

# affiche_dataloader(test_dataloader_v3)
```

[ ]:
```
model_inception_v3 = adapt_pretrained_model("inception_v3")
loss_fn = nn.CrossEntropyLoss()
optimiseur_inception_v3 = torch.optim.SGD(model_inception_v3.parameters(), lr=0.
 ↪001, momentum=0.9)
model_inception_v3_trainTime, model_inception_v3_trainModelComplexity,␣
 ↪model_inception_v3_validation_loss, model_inception_v3_accuracy, _ =␣
 ↪train(model_inception_v3, train_dataloader_v3, test_dataloader_v3, loss_fn,␣
 ↪optimiseur_inception_v3, epochs, typeTrain="inception_v3")
```

```
Downloading:
"https://download.pytorch.org/models/inception_v3_google-0cc3c7bd.pth" to
/root/.cache/torch/hub/checkpoints/inception_v3_google-0cc3c7bd.pth

  0%|          | 0.00/104M [00:00<?, ?B/s]

Device is : cuda:0


 ----------------------------------------
Epoch 1, Train Loss: 2.9687, Train Accuracy: 41.1245 %
         ----------------------------------------
        Predicted : tensor([ 7, 72, 82, 27,  2, 12, 50, 55, 38, 39, 23,  3,  5,
81,  3,  8, 98, 58,
        92, 16,  1, 16, 66, 76, 15, 89, 22, 98,  1, 40, 92, 30],
       device='cuda:0')
        Result : tensor([ 7, 72, 82, 59,  2, 12, 50, 55, 38, 39, 23,  3,  5, 81,
50,  8, 98, 58,
        92, 16,  1, 16, 66, 14, 15, 89, 22, 98,  1, 40, 92, 30],
       device='cuda:0')
        TOTAL (correct/total): 765.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.6447, Validation Accuracy: 88.5417 %
```

```
Epoch 3, Train Loss: 0.5172, Train Accuracy: 92.6101 %
         --------------------------------------
        Predicted : tensor([14,  3,  0,  5, 78,  1, 81, 16, 32,  1, 99, 46,  3,
5, 16, 57,  1,  0,
        50, 51, 72, 55, 94, 94, 98,  5, 94, 55, 96, 87,  2,  0],
       device='cuda:0')
        Result : tensor([14,  3,  0,  5, 78,  1, 81, 16, 32,  1, 99, 46,  3,  5,
16, 57,  1,  0,
        50, 51, 72, 79, 94, 94, 98,  5, 94, 79, 96, 87,  2,  0],
       device='cuda:0')
        TOTAL (correct/total): 827.0 / 864.0
         --------------------------------------
        Epoch 4, Validation Loss: 0.1746, Validation Accuracy: 95.7176 %
Epoch 5, Train Loss: 0.1566, Train Accuracy: 97.8099 %
         --------------------------------------
        Predicted : tensor([ 4,  3, 94,  5,  1, 63,  3, 20, 68,  1,  5, 34, 83,
5, 82, 23, 92, 93,
         5,  0, 66,  3, 27, 19, 86,  5, 79, 14,  3, 88, 21, 93],
       device='cuda:0')
        Result : tensor([ 4,  3, 94,  5,  1, 63,  3, 20, 68,  1,  5, 34, 70,  5,
82, 23, 92, 93,
         5,  0, 66,  3, 27, 19, 86,  5, 79, 14,  3, 88, 21, 93],
       device='cuda:0')
        TOTAL (correct/total): 838.0 / 864.0
         --------------------------------------
        Epoch 6, Validation Loss: 0.1213, Validation Accuracy: 96.9907 %
Epoch 7, Train Loss: 0.0802, Train Accuracy: 98.9114 %
         --------------------------------------
        Predicted : tensor([58,  3, 16, 11, 42, 23, 93, 13, 77,  3,  1,  3, 38,
1, 22, 23,  0, 72,
        77, 41, 91, 31, 57, 46, 19, 12, 16, 27,  8,  0,  1,  3],
       device='cuda:0')
        Result : tensor([58,  3, 16, 11, 42, 23, 93, 13, 77,  3,  1,  3, 38,  1,
22, 23,  0, 72,
        77, 41, 91, 31, 57, 46, 19, 12, 16, 27,  8,  0,  1,  3],
       device='cuda:0')
        TOTAL (correct/total): 840.0 / 864.0
         --------------------------------------
        Epoch 8, Validation Loss: 0.1177, Validation Accuracy: 97.2222 %

 --------------------------------------
```




```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
<IPython.core.display.Javascript object>
```

5.10 Comparaison des modèles

5.10.1 Pour : resnet18 VS alexnet

[ ]:
```
resultTest1 = compare_models(model_resnet18_trainTime,␣
  ↪model_resnet18_trainModelComplexity, model_resnet18_validation_loss,␣
  ↪model_resnet18_accuracy, model_alexnet_trainTime,␣
  ↪model_alexnet_trainModelComplexity, model_alexnet_validation_loss,␣
  ↪model_alexnet_accuracy)

if resultTest1 == 1:
```

```
    print("\n_____________________________________________________\nLe Modèle␣
 ↪resnet18 est le plus performant !
 ↪\n_____________________________________________________\n")
else:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪alexnet est le plus performant !
 ↪\n_____________________________________________________\n")
```

```
Le modèle 1 a une perte de 0.15122037219245815 sur les données de validation
avec une précision de 96.52777777777779% et une complexité de 11080357
paramètres et un temps d'évaluation de 380.01206827163696s

Le modèle 2 a une perte de 0.4305424852904657 sur les données de validation avec
une précision de 88.77314814814815% et une complexité de 57417637 paramètres et
un temps d'évaluation de 321.05346512794495s

        Le modèle 1 a une perte plus faible sur les données de validation.


-----------------------------------------------------
Le Modèle resnet18 est le plus performant !
-----------------------------------------------------
```

5.10.2 Pour : squeezenet1_0 VS vgg16

[ ]:
```
resultTest1 = compare_models(model_squeezenet1_0_trainTime,␣
 ↪model_squeezenet1_0_trainModelComplexity,␣
 ↪model_squeezenet1_0_validation_loss, model_squeezenet1_0_accuracy,␣
 ↪model_vgg16_trainTime, model_vgg16_trainModelComplexity,␣
 ↪model_vgg16_validation_loss, model_vgg16_accuracy)

if resultTest1 == 1:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪squeezenet1_0 est le plus performant !
 ↪\n_____________________________________________________\n")
else:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪vgg16 est le plus performant !
 ↪\n_____________________________________________________\n")
```

```
Le modèle 1 a une perte de 0.4824790313098304 sur les données de validation avec
une précision de 89.00462962962963% et une complexité de 787237 paramètres et un
temps d'évaluation de 373.0097849369049s

Le modèle 2 a une perte de 0.199863223909132 sur les données de validation avec
une précision de 94.9074074074074% et une complexité de 134674341 paramètres et
un temps d'évaluation de 1184.215722322464s
```

Le modèle 2 a une perte plus faible sur les données de validation.

```
--------------------------------------------------
Le Modèle vgg16 est le plus performant !
--------------------------------------------------
```

5.10.3 Pour : alexnet VS squeezenet1_0

[ ]:
```
resultTest1 = compare_models(model_alexnet_trainTime,␣
 ↪model_alexnet_trainModelComplexity, model_alexnet_validation_loss,␣
 ↪model_alexnet_accuracy, model_squeezenet1_0_trainTime,␣
 ↪model_squeezenet1_0_trainModelComplexity,␣
 ↪model_squeezenet1_0_validation_loss, model_squeezenet1_0_accuracy)

if resultTest1 == 1:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪alexnet est le plus performant !
 ↪\n_____________________________________________________\n")
else:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪squeezenet1_0 est le plus performant !
 ↪\n_____________________________________________________\n")
```

Le modèle 1 a une perte de 0.4305424852904657 sur les données de validation avec une précision de 88.77314814814815% et une complexité de 57417637 paramètres et un temps d'évaluation de 321.05346512794495s

Le modèle 2 a une perte de 0.4824790313098304 sur les données de validation avec une précision de 89.00462962962963% et une complexité de 787237 paramètres et un temps d'évaluation de 373.0097849369049s

Le modèle 1 a une perte plus faible sur les données de validation.

```
--------------------------------------------------
Le Modèle alexnet est le plus performant !
--------------------------------------------------
```

5.10.4 Pour : alexnet VS vgg16

[ ]:
```
resultTest1 = compare_models(model_alexnet_trainTime,␣
 ↪model_alexnet_trainModelComplexity, model_alexnet_validation_loss,␣
 ↪model_alexnet_accuracy, model_vgg16_trainTime,␣
 ↪model_vgg16_trainModelComplexity, model_vgg16_validation_loss,␣
 ↪model_vgg16_accuracy)

if resultTest1 == 1:
```

```
    print("\n_____________________________________________________\nLe Modèle␣
 ↪alexnet est le plus performant !
 ↪\n_____________________________________________________\n")
else:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪vgg16 est le plus performant !
 ↪\n_____________________________________________________\n")
```

```
Le modèle 1 a une perte de 0.4305424852904657 sur les données de validation avec
une précision de 88.77314814814815% et une complexité de 57417637 paramètres et
un temps d'évaluation de 321.05346512794495s

Le modèle 2 a une perte de 0.199863223909132 sur les données de validation avec
une précision de 94.9074074074074% et une complexité de 134674341 paramètres et
un temps d'évaluation de 1184.215722322464s

        Le modèle 2 a une perte plus faible sur les données de validation.


-----------------------------------------------------
Le Modèle vgg16 est le plus performant !
-----------------------------------------------------
```

5.10.5 Pour : resnet18 VS vgg16

[ ]:
```
resultTest1 = compare_models(model_resnet18_trainTime,␣
 ↪model_resnet18_trainModelComplexity, model_resnet18_validation_loss,␣
 ↪model_resnet18_accuracy, model_vgg16_trainTime,␣
 ↪model_vgg16_trainModelComplexity, model_vgg16_validation_loss,␣
 ↪model_vgg16_accuracy)

if resultTest1 == 1:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪resnet18 est le plus performant !
 ↪\n_____________________________________________________\n")
else:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪vgg16 est le plus performant !
 ↪\n_____________________________________________________\n")
```

```
Le modèle 1 a une perte de 0.15122037219245815 sur les données de validation
avec une précision de 96.52777777777779% et une complexité de 11080357
paramètres et un temps d'évaluation de 380.01206827163696s

Le modèle 2 a une perte de 0.199863223909132 sur les données de validation avec
une précision de 94.9074074074074% et une complexité de 134674341 paramètres et
un temps d'évaluation de 1184.215722322464s
```

```
        Le modèle 1 a une perte plus faible sur les données de validation.

--------------------------------------------------
Le Modèle resnet18 est le plus performant !
--------------------------------------------------
```

5.10.6 Pour : resnet18 VS densenet161

[ ]:
```
resultTest1 = compare_models(model_resnet18_trainTime,␣
 ↪model_resnet18_trainModelComplexity, model_resnet18_validation_loss,␣
 ↪model_resnet18_accuracy, model_densenet161_trainTime,␣
 ↪model_densenet161_trainModelComplexity, model_densenet161_validation_loss,␣
 ↪model_densenet161_accuracy)

if resultTest1 == 1:
    print("\n_____\nLe Modèle␣
 ↪resnet18 est le plus performant !
 ↪\n_____\n")
else:
    print("\n_____\nLe Modèle␣
 ↪densenet161 est le plus performant !
 ↪\n_____\n")
```

```
Le modèle 1 a une perte de 0.15122037219245815 sur les données de validation
avec une précision de 96.52777777777779% et une complexité de 11080357
paramètres et un temps d'évaluation de 380.01206827163696s

Le modèle 2 a une perte de 0.0989958343195246 sur les données de validation avec
une précision de 97.45370370370371% et une complexité de 26695109 paramètres et
un temps d'évaluation de 1690.681633234024s

        Le modèle 2 a une perte plus faible sur les données de validation.

--------------------------------------------------
Le Modèle densenet161 est le plus performant !
--------------------------------------------------
```

5.10.7 Pour : densenet161 VS inception_v3

[ ]:
```
resultTest1 = compare_models(model_densenet161_trainTime,␣
 ↪model_densenet161_trainModelComplexity, model_densenet161_validation_loss,␣
 ↪model_densenet161_accuracy, model_inception_v3_trainTime,␣
 ↪model_inception_v3_trainModelComplexity, model_inception_v3_validation_loss,␣
 ↪model_inception_v3_accuracy)

if resultTest1 == 1:
```

```
    print("\n_____________________________________________________\nLe Modèle␣
 ↪densenet161 est le plus performant !
 ↪\n_____________________________________________________\n")
else:
    print("\n_____________________________________________________\nLe Modèle␣
 ↪inception_v3 est le plus performant !
 ↪\n_____________________________________________________\n")
```

```
Le modèle 1 a une perte de 0.0989958343195246 sur les données de validation avec
une précision de 97.45370370370371% et une complexité de 26695109 paramètres et
un temps d'évaluation de 1690.681633234024s

Le modèle 2 a une perte de 0.11765296857252165 sur les données de validation
avec une précision de 97.22222222222221% et une complexité de 25319213
paramètres et un temps d'évaluation de 1174.6039533615112s

        Le modèle 1 a une perte plus faible sur les données de validation.


-----------------------------------------------------
Le Modèle densenet161 est le plus performant !
-----------------------------------------------------
```

6 Exécution des boucles d'apprentissage pour k itération

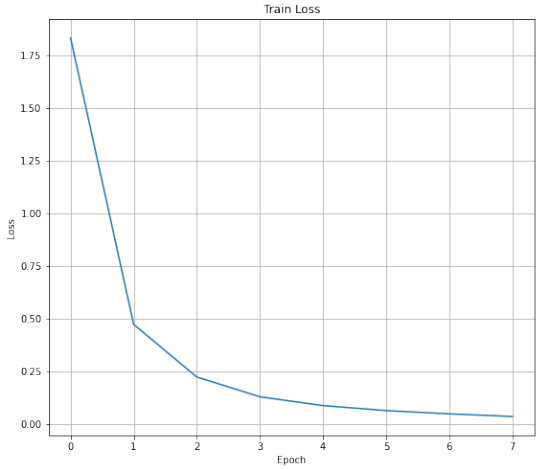
[14]:
```
def boucle_de_validation_croisee_pour_k_iteration(model, train_loader,␣
 ↪validation_loader, loss_fn, optimizer, epochs, typeTrain="deepLearning",␣
 ↪k=5):
    scores_total = []
    for i in range(k):
        # train_dataloader, test_dataloader = prepare_data(mean, std,␣
 ↪size_of_image, transform, batch_size, split)
        train_dataloader, test_dataloader = prepare_data_on_google_colab(mean,␣
 ↪std, size_of_image, transform, batch_size, split)

        # Apprentissage des données sur le Train et la Validation
        _, _, _, _, scores = train(model, train_dataloader, test_dataloader,␣
 ↪loss_fn, optimizer, epochs, typeTrain=typeTrain)
        scores_total += scores

    # Calcul de la moyenne des scores sur les k itérations
    mean_score = sum(scores_total) / k
    print("La moyenne des accuracy: {:.4f} %".format(mean_score * 100.0))
    return mean_score
```

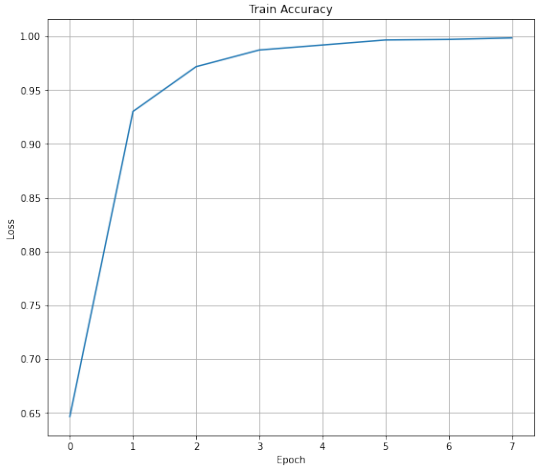
6.1 Pour le modèle resnet18 avec K = 5

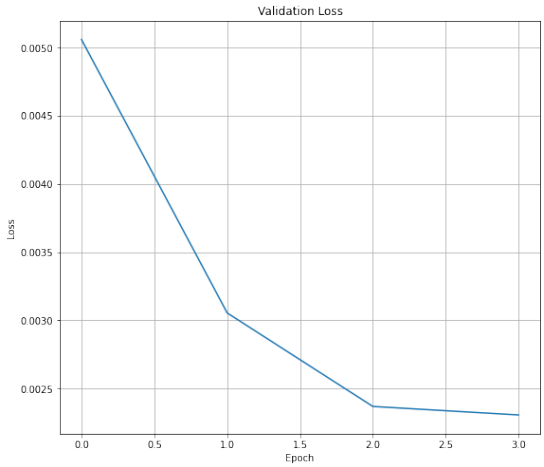
```
[ ]: model_resnet18 = adapt_pretrained_model("resnet18")
loss_fn = nn.CrossEntropyLoss()
optimiseur_resnet18 = torch.optim.SGD(model_resnet18.parameters(), lr=0.001,␣
 ↪momentum=0.9)

mean_resnet18 = boucle_de_validation_croisee_pour_k_iteration(model_resnet18,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_resnet18, epochs,␣
 ↪typeTrain="resnet18")
```

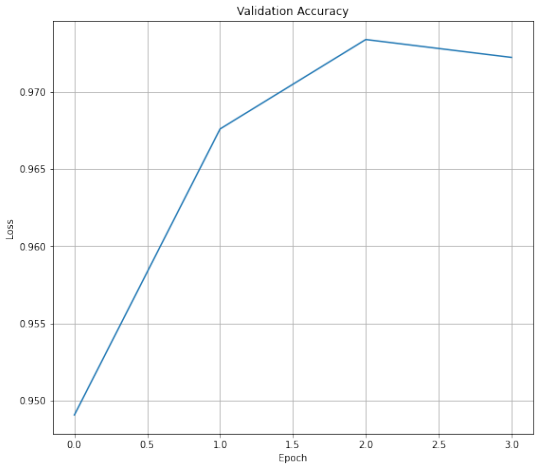
```
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to
/root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth

  0%|          | 0.00/44.7M [00:00<?, ?B/s]

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0


 ----------------------------------------
Epoch 1, Train Loss: 1.8332, Train Accuracy: 64.6644 %
         ----------------------------------------
        Predicted : tensor([ 5, 26, 50, 81, 50, 77,  3,  1, 38,  5, 60, 85, 35,
0,  1, 83],
       device='cuda:0')
        Result : tensor([ 5, 26, 50, 81, 50, 77,  3,  1, 38,  5, 60, 85, 35,  0,
1, 16],
       device='cuda:0')
        TOTAL (correct/total): 820.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.2732, Validation Accuracy: 94.9074 %
Epoch 3, Train Loss: 0.2227, Train Accuracy: 97.1824 %
         ----------------------------------------
        Predicted : tensor([ 3,  3,  5, 94, 57, 23, 49, 76, 18, 41,  2, 46, 23,
38,  3, 12],
       device='cuda:0')
        Result : tensor([ 3,  3,  5, 94, 57, 23, 49, 76, 18, 41,  2, 46, 23, 22,
3, 12],
       device='cuda:0')
        TOTAL (correct/total): 836.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.1649, Validation Accuracy: 96.7593 %
Epoch 5, Train Loss: 0.0869, Train Accuracy: 99.1931 %
         ----------------------------------------
        Predicted : tensor([ 23,   1,  40,  23,  52,   0,  12,   5,  65,   2,
3, 100,   0,  34,
         5,   3], device='cuda:0')
        Result : tensor([ 23,   1,  40,  23,  52,   0,  12,   5,  65,   2,   3,
```

```
100,   0,  34,
          5,   3], device='cuda:0')
         TOTAL (correct/total): 841.0 / 864.0
          ----------------------------------------
         Epoch 6, Validation Loss: 0.1279, Validation Accuracy: 97.3380 %
Epoch 7, Train Loss: 0.0476, Train Accuracy: 99.7182 %
         ----------------------------------------
         Predicted : tensor([84, 82, 40, 33,  5, 12, 68, 27, 19, 36, 50, 25,  3,
41, 47,  3],
       device='cuda:0')
         Result : tensor([84, 82, 40, 33,  5, 12, 68, 27, 19, 36, 50, 25,  3, 41,
47,  3],
       device='cuda:0')
         TOTAL (correct/total): 840.0 / 864.0
          ----------------------------------------
         Epoch 8, Validation Loss: 0.1245, Validation Accuracy: 97.2222 %

 ----------------------------------------
```



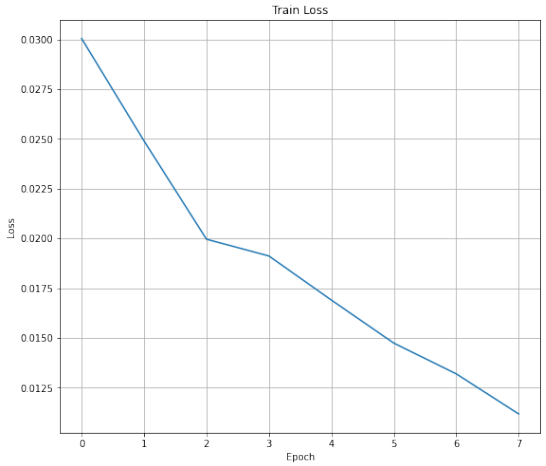


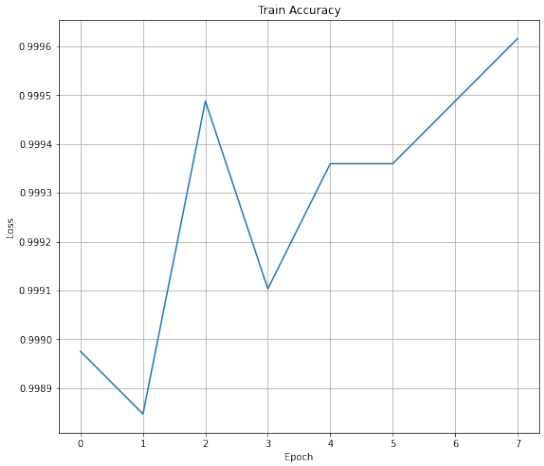


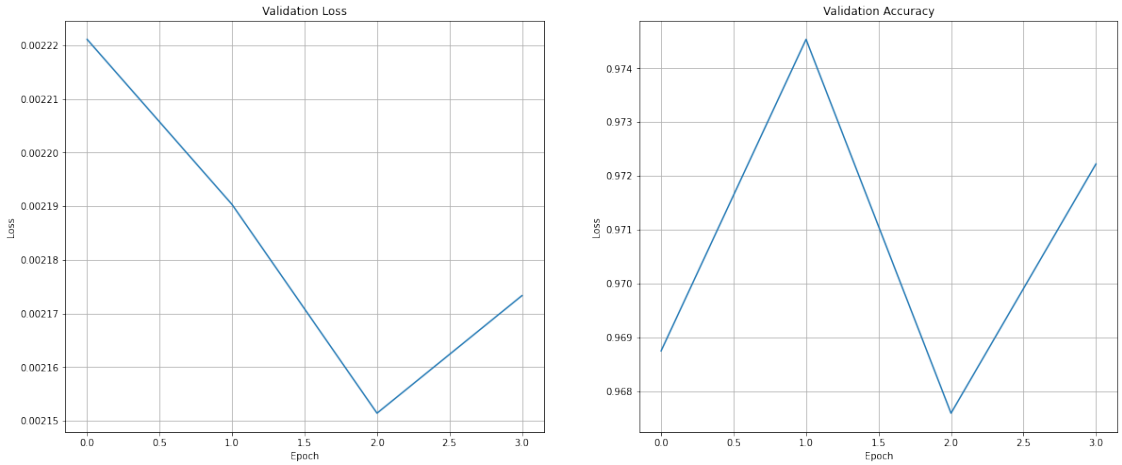



```
<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0300, Train Accuracy: 99.8975 %
         ----------------------------------------
        Predicted : tensor([ 39,  43,   5,  14,   3,   3,   1,  87,  19,   0,
71,   5,  25,  39,
        100,  41], device='cuda:0')
        Result : tensor([39, 43,  5, 14,  3,  3,  1, 87, 19,  0, 71,  5, 25, 39,
17, 41],
       device='cuda:0')
        TOTAL (correct/total): 837.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.1199, Validation Accuracy: 96.8750 %
Epoch 3, Train Loss: 0.0200, Train Accuracy: 99.9488 %
         ----------------------------------------
        Predicted : tensor([ 5, 73,  0, 94, 50,  3,  7, 26, 41,  0, 19,  0, 94,
5, 12, 82],
       device='cuda:0')
        Result : tensor([ 5, 73,  0, 94, 50,  3,  7, 26, 41,  0, 19,  0, 94,  5,
12, 82],
       device='cuda:0')
        TOTAL (correct/total): 842.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.1183, Validation Accuracy: 97.4537 %
Epoch 5, Train Loss: 0.0169, Train Accuracy: 99.9360 %
         ----------------------------------------
        Predicted : tensor([94, 76,  3, 45, 17, 67,  2,  1, 65,  3, 39,  0, 96,
```

```
16, 75, 40],
       device='cuda:0')
        Result : tensor([94, 76,  3, 45, 17, 67,  2,  1, 65,  3, 39,  0, 96, 16,
75, 40],
       device='cuda:0')
        TOTAL (correct/total): 836.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.1162, Validation Accuracy: 96.7593 %
Epoch 7, Train Loss: 0.0132, Train Accuracy: 99.9488 %
         ----------------------------------------
        Predicted : tensor([92,  5, 13,  5, 24,  3, 41, 33, 19,  6,  5, 23, 54,
81, 93, 40],
       device='cuda:0')
        Result : tensor([92,  5, 13,  5, 24,  3, 41, 33, 19,  6,  5, 23, 54, 81,
93, 40],
       device='cuda:0')
        TOTAL (correct/total): 840.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.1174, Validation Accuracy: 97.2222 %

 ----------------------------------------
```

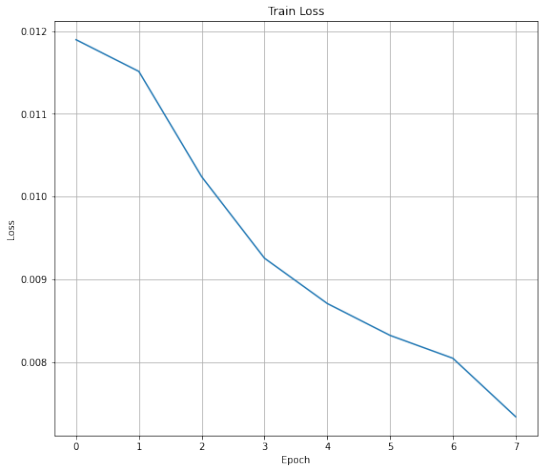


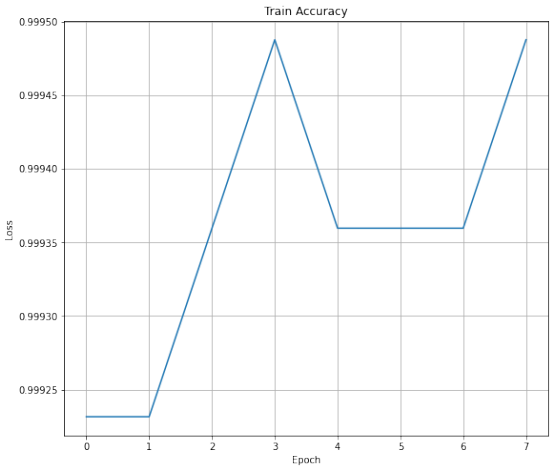


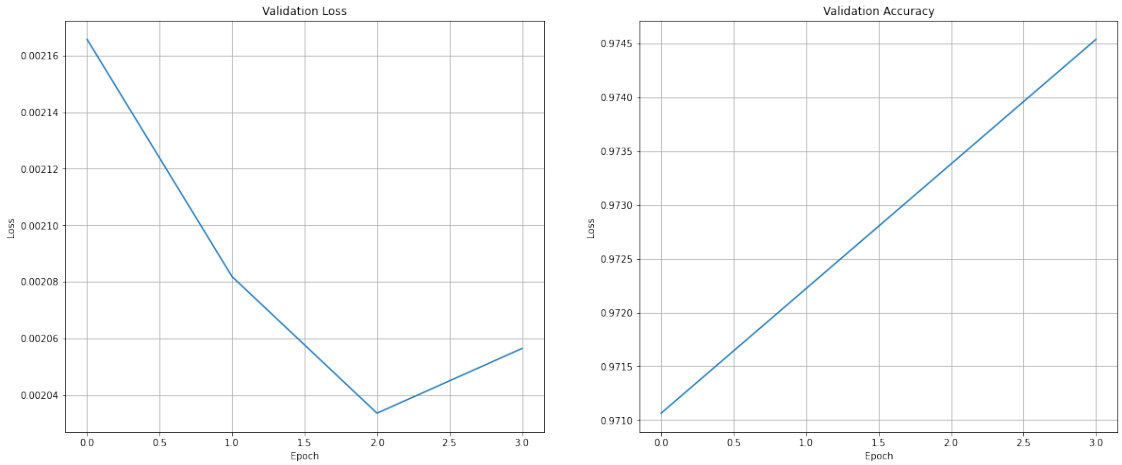



```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 0:06:20 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0119, Train Accuracy: 99.9232 %
         ----------------------------------------
        Predicted : tensor([37, 87, 90, 56, 94,  0, 27,  3,  5, 91, 23, 49, 26,
39,  0, 52],
       device='cuda:0')
        Result : tensor([37, 87, 90, 56, 94,  0, 27,  3,  5, 91, 23, 49, 26, 39,
0, 52],
       device='cuda:0')
        TOTAL (correct/total): 839.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.1169, Validation Accuracy: 97.1065 %
Epoch 3, Train Loss: 0.0102, Train Accuracy: 99.9360 %
         ----------------------------------------
        Predicted : tensor([82,  3, 16, 87, 20,  3,  1, 75, 94, 56, 79, 98, 49,
14, 66, 23],
       device='cuda:0')
        Result : tensor([82,  3, 16, 87, 20,  3,  1, 75, 94, 56, 79, 98, 49, 14,
66, 23],
       device='cuda:0')
        TOTAL (correct/total): 840.0 / 864.0
```

```
 ----------------------------------------
Epoch 4, Validation Loss: 0.1124, Validation Accuracy: 97.2222 %
Epoch 5, Train Loss: 0.0087, Train Accuracy: 99.9360 %
 ----------------------------------------
Predicted : tensor([75, 87,  3, 37,  5, 56, 26, 56, 47, 15, 34, 41, 23,
56,  0, 19],
      device='cuda:0')
Result : tensor([75, 87,  3, 37,  5, 56, 26, 56, 47, 15, 34, 41, 15, 56,
0, 19],
      device='cuda:0')
TOTAL (correct/total): 841.0 / 864.0
 ----------------------------------------
Epoch 6, Validation Loss: 0.1098, Validation Accuracy: 97.3380 %
Epoch 7, Train Loss: 0.0080, Train Accuracy: 99.9360 %
 ----------------------------------------
Predicted : tensor([12, 34, 16,  0, 69, 74,  5, 77, 41, 54,  0, 71, 12,
39, 12,  5],
      device='cuda:0')
Result : tensor([12, 34, 16,  0, 69, 74,  5, 77, 41, 54,  0, 71, 12, 39,
12,  5],
      device='cuda:0')
TOTAL (correct/total): 842.0 / 864.0
 ----------------------------------------
Epoch 8, Validation Loss: 0.1110, Validation Accuracy: 97.4537 %

----------------------------------------
```

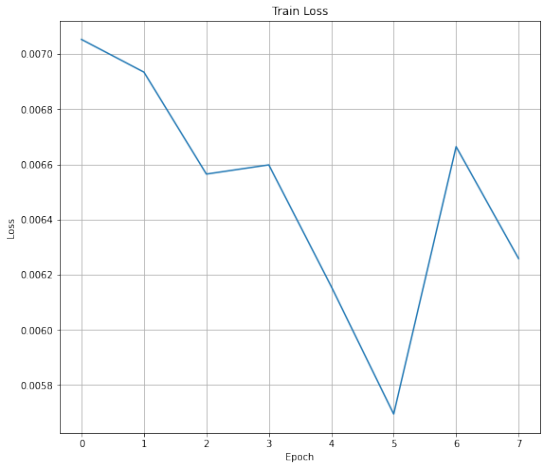


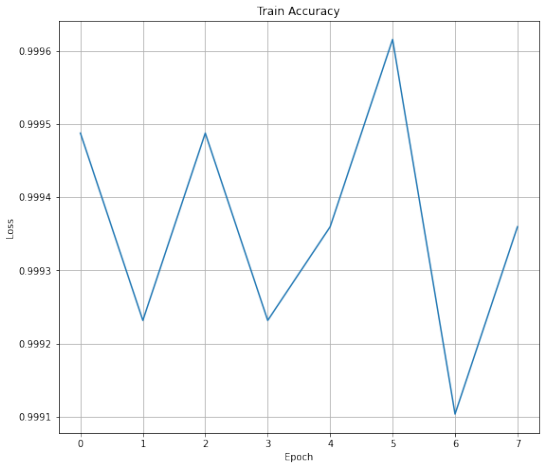


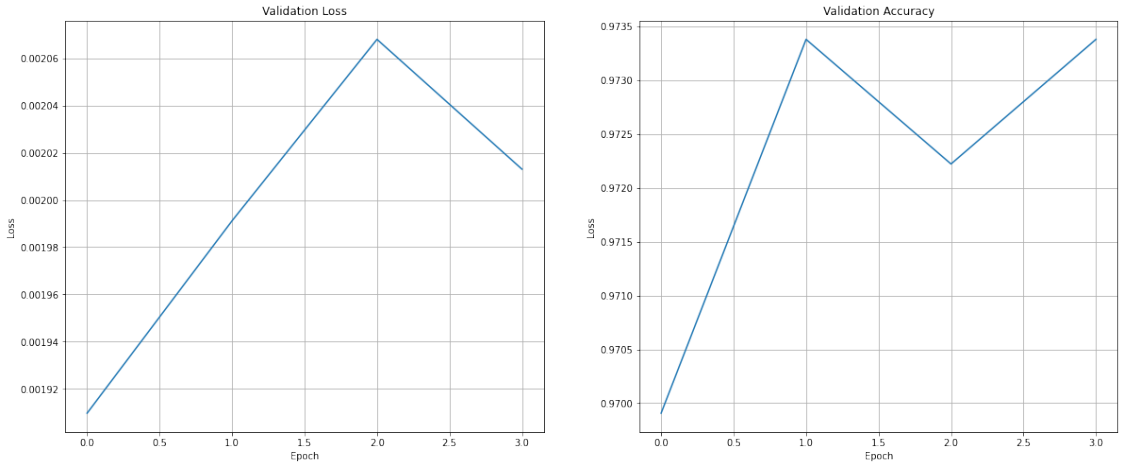



```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 0:12:39 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0071, Train Accuracy: 99.9488 %
         ----------------------------------------
        Predicted : tensor([ 46,  19,  57, 100, 100,  75,  47,  20,  16,  82,
1,  75,  56,   9,
         94,  68], device='cuda:0')
        Result : tensor([ 46,  19,  57, 100,  17,  75,  47,  20,  16,  82,   1,
75,  56,   9,
         94,  68], device='cuda:0')
        TOTAL (correct/total): 838.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.1031, Validation Accuracy: 96.9907 %
Epoch 3, Train Loss: 0.0066, Train Accuracy: 99.9488 %
         ----------------------------------------
        Predicted : tensor([72, 75, 15, 25, 86,  3, 12,  3, 67, 36, 87, 94,  3,
41, 46, 50],
       device='cuda:0')
        Result : tensor([72, 58, 15, 25, 86,  3, 12,  3, 67, 36, 87, 94,  3, 41,
46, 50],
       device='cuda:0')
        TOTAL (correct/total): 841.0 / 864.0
```

```
 ---------------------------------------
Epoch 4, Validation Loss: 0.1075, Validation Accuracy: 97.3380 %
Epoch 5, Train Loss: 0.0062, Train Accuracy: 99.9360 %
 ---------------------------------------
Predicted : tensor([29, 19, 19, 20, 75, 47, 25, 91, 60,  5, 16,  3,  3,
10,  1, 96],
       device='cuda:0')
Result : tensor([29, 19, 19, 20, 75, 47, 25, 91, 95,  5, 16,  3,  3, 10,
1, 96],
       device='cuda:0')
TOTAL (correct/total): 840.0 / 864.0
 ---------------------------------------
Epoch 6, Validation Loss: 0.1117, Validation Accuracy: 97.2222 %
Epoch 7, Train Loss: 0.0067, Train Accuracy: 99.9103 %
 ---------------------------------------
Predicted : tensor([80, 49, 64, 47,  3, 94, 10,  5, 26, 78,  0, 75,  5,
3, 69, 14],
       device='cuda:0')
Result : tensor([80, 49, 64, 47,  3, 94, 10,  5, 26, 78,  0, 75,  5,  3,
69, 14],
       device='cuda:0')
TOTAL (correct/total): 841.0 / 864.0
 ---------------------------------------
Epoch 8, Validation Loss: 0.1087, Validation Accuracy: 97.3380 %

 ----------------------------------------
```

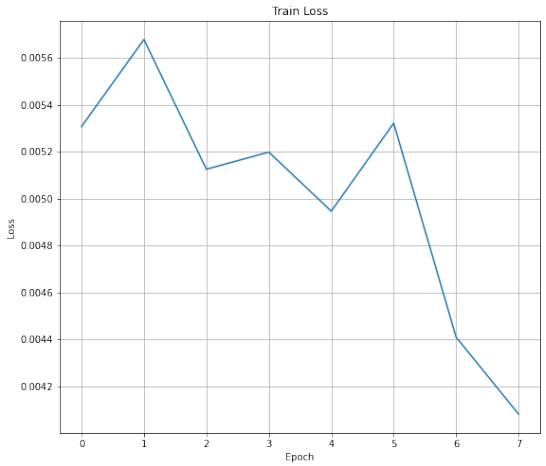


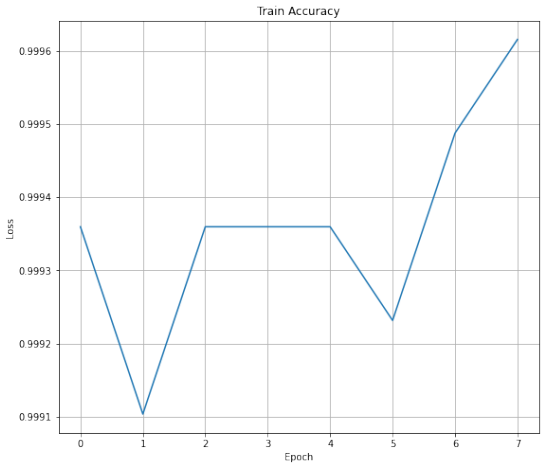


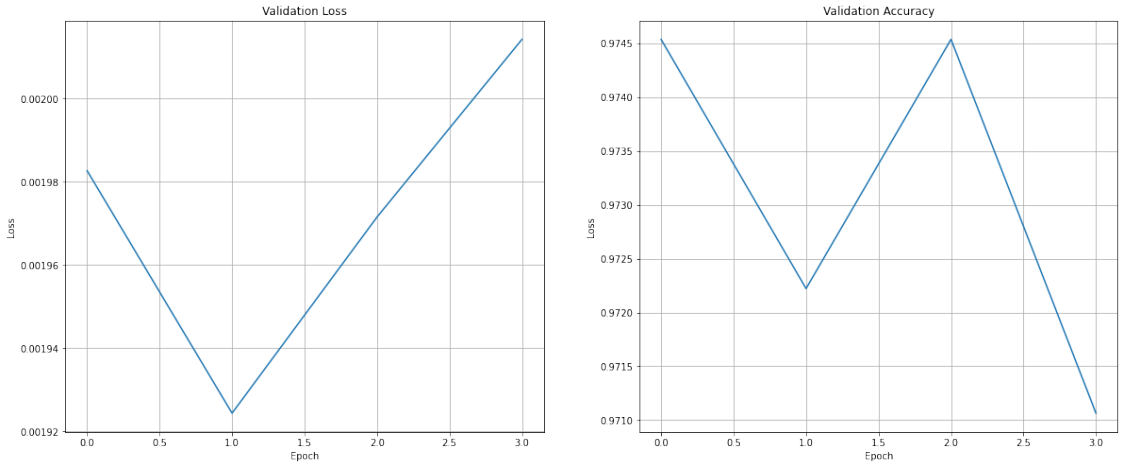



```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 0:18:56 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0053, Train Accuracy: 99.9360 %
         ----------------------------------------
        Predicted : tensor([98, 50,  5,  1, 76, 30,  5,  5, 14, 14, 80, 39, 62,
87, 43,  0],
       device='cuda:0')
        Result : tensor([98, 50,  5,  1, 76, 30,  5,  5, 14, 14, 80, 39, 62, 87,
43,  0],
       device='cuda:0')
        TOTAL (correct/total): 842.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.1071, Validation Accuracy: 97.4537 %
Epoch 3, Train Loss: 0.0051, Train Accuracy: 99.9360 %
         ----------------------------------------
        Predicted : tensor([69, 94, 48, 15,  3,  5, 96,  1, 34,  3, 21, 94,  5,
12, 19,  3],
       device='cuda:0')
        Result : tensor([69, 94, 48, 15,  3,  5, 96,  1, 34,  3, 21, 94,  5, 12,
19,  3],
       device='cuda:0')
        TOTAL (correct/total): 840.0 / 864.0
```

```
 ----------------------------------------
 Epoch 4, Validation Loss: 0.1039, Validation Accuracy: 97.2222 %
Epoch 5, Train Loss: 0.0049, Train Accuracy: 99.9360 %
 ----------------------------------------
 Predicted : tensor([89, 33,  3,  3, 48, 49,  3, 45, 71, 13, 13,  3, 75,
1, 63, 25],
       device='cuda:0')
 Result : tensor([89, 33,  3,  3, 48, 49,  3, 45, 71, 13, 13,  3, 75,  1,
63, 25],
       device='cuda:0')
 TOTAL (correct/total): 842.0 / 864.0
 ----------------------------------------
 Epoch 6, Validation Loss: 0.1065, Validation Accuracy: 97.4537 %
Epoch 7, Train Loss: 0.0044, Train Accuracy: 99.9488 %
 ----------------------------------------
 Predicted : tensor([ 2, 36,  0, 54, 98, 47,  3, 76, 23, 91, 25, 39, 46,
3, 64,  5],
       device='cuda:0')
 Result : tensor([ 2, 36,  0, 18, 98, 47,  3, 76, 23, 91, 25, 39, 46,  3,
64,  5],
       device='cuda:0')
 TOTAL (correct/total): 839.0 / 864.0
 ----------------------------------------
 Epoch 8, Validation Loss: 0.1088, Validation Accuracy: 97.1065 %

----------------------------------------
```








```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 2844), started 0:25:13 ago. (Use '!kill␣
 ↪2844' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

```
La moyenne des accuracy: 388.3565 %
```

6.2 Pour le modèle alexnet avec K = 5

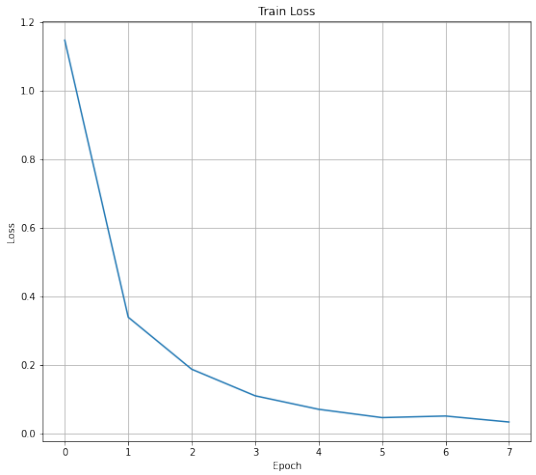
[ ]:
```
model_alexnet = adapt_pretrained_model("alexnet")
loss_fn = nn.CrossEntropyLoss()
optimiseur_alexnet = torch.optim.SGD(model_alexnet.parameters(), lr=0.001,␣
 ↪momentum=0.9)

mean_alexnet = boucle_de_validation_croisee_pour_k_iteration(model_alexnet,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_alexnet, epochs,␣
 ↪typeTrain="alexnet")
```

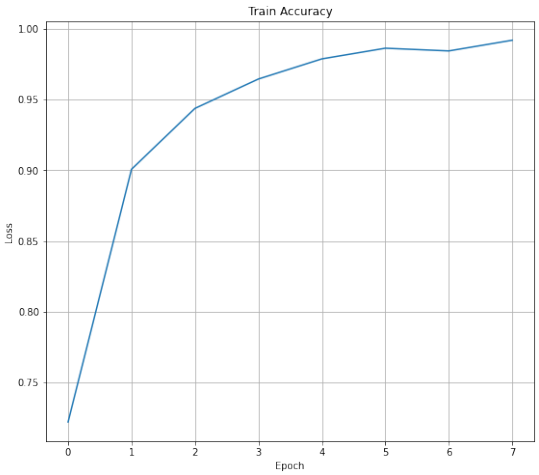
```
Downloading: "https://download.pytorch.org/models/alexnet-owt-7be5be79.pth" to
/root/.cache/torch/hub/checkpoints/alexnet-owt-7be5be79.pth
```

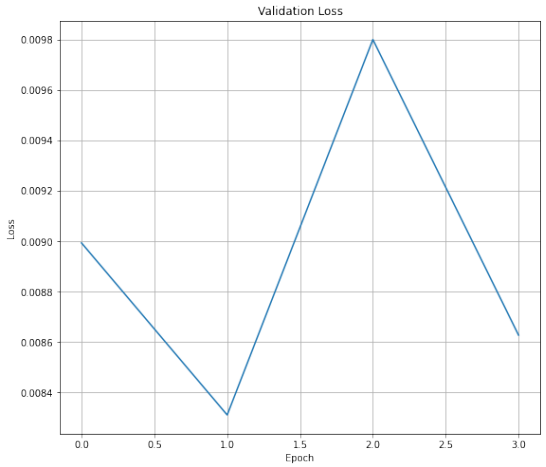
```
  0%|          | 0.00/233M [00:00<?, ?B/s]
```

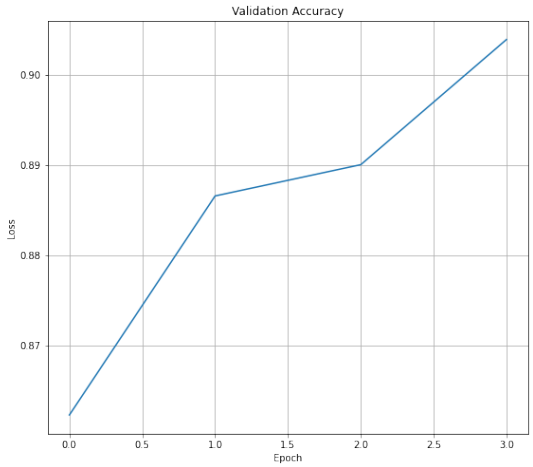
```
La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 1.1460, Train Accuracy: 72.2080 %
         ----------------------------------------
        Predicted : tensor([15, 63, 80, 67, 79,  5, 58, 16, 55, 39, 68, 19, 31,
```

```
16, 94, 19],
       device='cuda:0')
        Result : tensor([15, 63, 80, 91, 55,  5, 55, 16, 55, 39, 68, 19, 31, 16,
94, 19],
       device='cuda:0')
        TOTAL (correct/total): 745.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.4857, Validation Accuracy: 86.2269 %
Epoch 3, Train Loss: 0.1871, Train Accuracy: 94.3776 %
         ---------------------------------------
        Predicted : tensor([ 2, 44,  1,  3, 56, 12,  0, 23, 49, 99, 37,  0, 24,
55, 36, 39],
       device='cuda:0')
        Result : tensor([ 2, 44,  1,  3, 56, 12,  0, 23, 49,  6, 37,  0, 58, 55,
36, 39],
       device='cuda:0')
        TOTAL (correct/total): 766.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.4488, Validation Accuracy: 88.6574 %
Epoch 5, Train Loss: 0.0704, Train Accuracy: 97.8740 %
         ---------------------------------------
        Predicted : tensor([57,  0, 92,  3, 69, 65,  3,  0, 55,  3, 81,  2, 88,
80,  5, 60],
       device='cuda:0')
        Result : tensor([57,  0, 92,  3, 69, 65,  3,  0, 55,  3, 81,  2, 88, 80,
5, 60],
       device='cuda:0')
        TOTAL (correct/total): 769.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.5293, Validation Accuracy: 89.0046 %
Epoch 7, Train Loss: 0.0508, Train Accuracy: 98.4375 %
         ---------------------------------------
        Predicted : tensor([56, 88, 56, 58,  3, 36,  1, 33, 19,  6, 94, 81, 41,
65, 23,  2],
       device='cuda:0')
        Result : tensor([56, 88, 56, 58,  3, 36,  1, 33, 19,  6, 94, 81, 41, 65,
23,  2],
       device='cuda:0')
        TOTAL (correct/total): 781.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.4659, Validation Accuracy: 90.3935 %

 ---------------------------------------
```



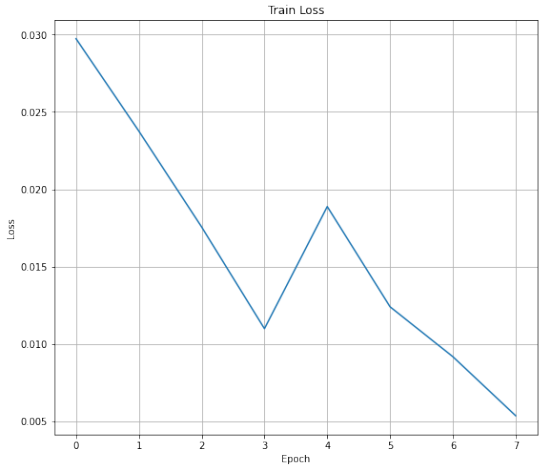


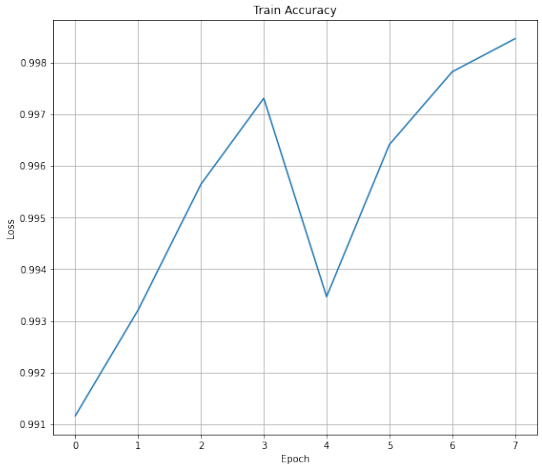


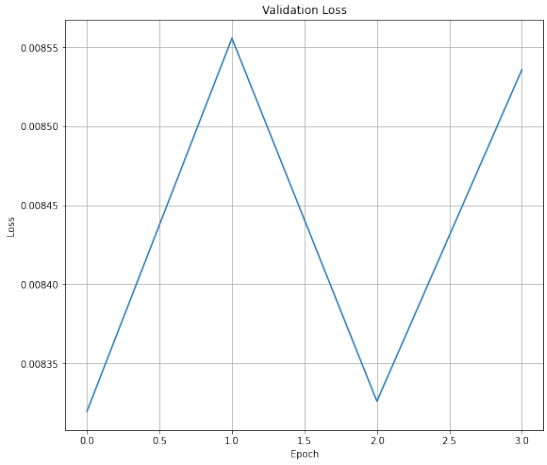


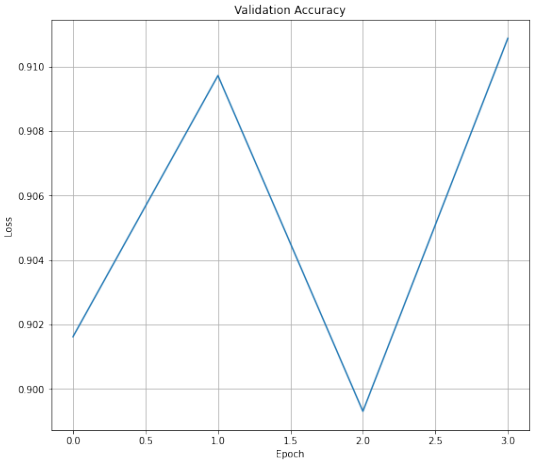

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 0:30:43 ago. (Use '!kill␣
  ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0297, Train Accuracy: 99.1163 %
         ----------------------------------------
        Predicted : tensor([ 3, 55, 94,  2,  0, 54, 15,  3, 92, 21, 76,  1,  3,
```

```
56,  5, 82],
       device='cuda:0')
        Result : tensor([ 3, 55, 94,  2,  0, 54, 15,  3, 92, 21, 76,  1,  3, 56,
5, 82],
       device='cuda:0')
        TOTAL (correct/total): 779.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.4493, Validation Accuracy: 90.1620 %
Epoch 3, Train Loss: 0.0176, Train Accuracy: 99.5645 %
         ---------------------------------------
        Predicted : tensor([ 5,  0, 12,  9, 18,  2,  5, 84, 85,  5,  3, 11,  4,
3, 47,  1],
       device='cuda:0')
        Result : tensor([ 5,  0, 12,  9, 18,  2,  5, 84, 85,  5,  3, 45,  4,  3,
67,  1],
       device='cuda:0')
        TOTAL (correct/total): 786.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.4620, Validation Accuracy: 90.9722 %
Epoch 5, Train Loss: 0.0189, Train Accuracy: 99.3468 %
         ---------------------------------------
        Predicted : tensor([53, 50, 55,  0, 86,  5,  5, 84, 18, 33,  5,  5,  1,
50,  1,  1],
       device='cuda:0')
        Result : tensor([53, 50, 55,  0, 86,  5,  5, 84, 18, 33,  5,  5,  1, 50,
1,  1],
       device='cuda:0')
        TOTAL (correct/total): 777.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.4496, Validation Accuracy: 89.9306 %
Epoch 7, Train Loss: 0.0092, Train Accuracy: 99.7823 %
         ---------------------------------------
        Predicted : tensor([20,  3, 27, 48, 63,  5, 58,  5, 61, 38, 42, 51, 53,
10,  3, 94],
       device='cuda:0')
        Result : tensor([20,  3, 27, 48, 63,  5, 54,  5, 61, 38, 42, 51, 53, 10,
3, 94],
       device='cuda:0')
        TOTAL (correct/total): 787.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.4609, Validation Accuracy: 91.0880 %

 ---------------------------------------
```



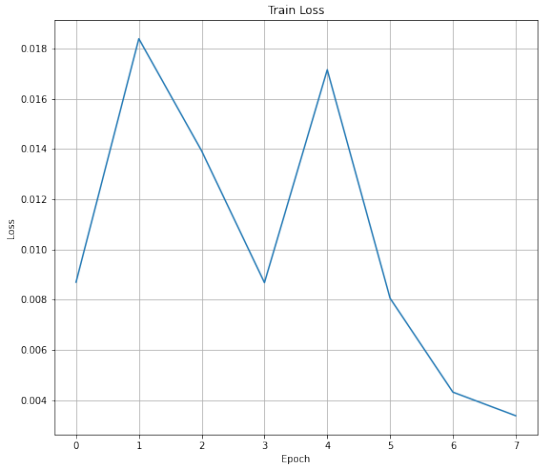


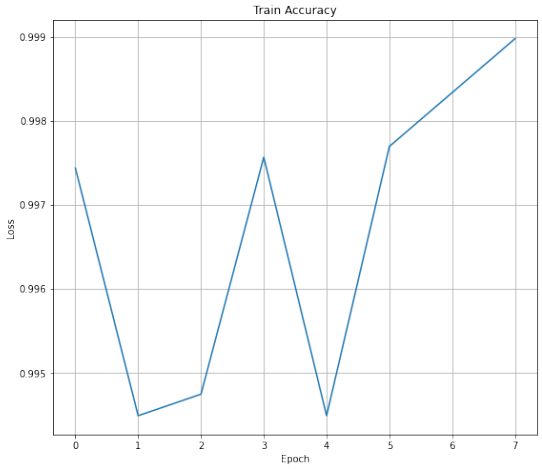


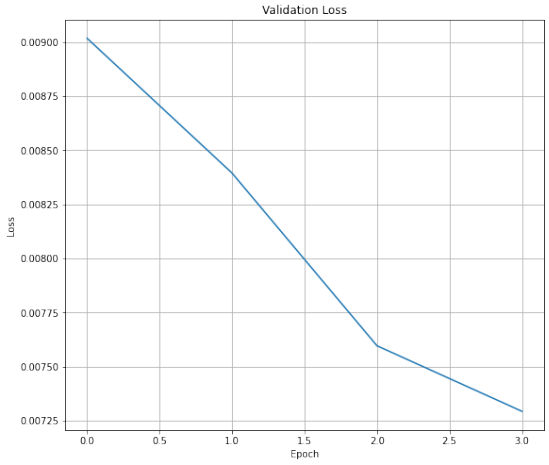


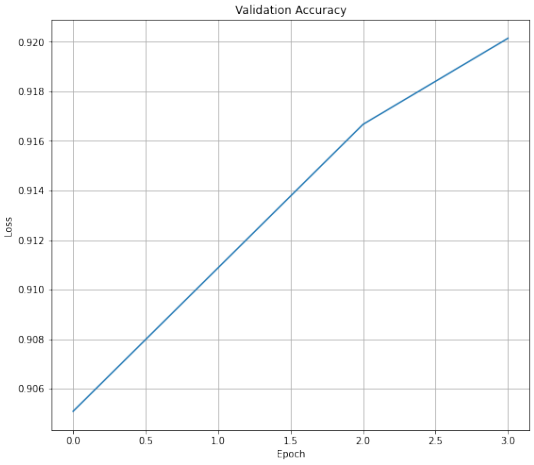

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 0:36:01 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0087, Train Accuracy: 99.7439 %
        ----------------------------------------
       Predicted : tensor([46, 63,  6,  3, 57, 98, 86, 50,  5,  5, 25,  5,  2,
71,  2, 21],
```

```
       device='cuda:0')
        Result : tensor([46, 66,  6,  3, 57, 98,  6, 50,  5,  5, 25,  5,  2, 71,
2, 21],
       device='cuda:0')
        TOTAL (correct/total): 782.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.4870, Validation Accuracy: 90.5093 %
Epoch 3, Train Loss: 0.0139, Train Accuracy: 99.4749 %
         ---------------------------------------
        Predicted : tensor([ 0,  86,  46,  72,   2, 100,  83,  19,  91,  55,
12,  39,  19,  23,
         55,  87], device='cuda:0')
        Result : tensor([ 0,   6,  46,  72,   2, 100,  83,  19,  91,  55,  12,
39,  19,  23,
         55,  87], device='cuda:0')
        TOTAL (correct/total): 787.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.4533, Validation Accuracy: 91.0880 %
Epoch 5, Train Loss: 0.0172, Train Accuracy: 99.4493 %
         ---------------------------------------
        Predicted : tensor([28, 47, 18, 97,  3, 40, 15,  1, 90,  2,  5, 41, 12,
52, 75, 35],
       device='cuda:0')
        Result : tensor([44, 47, 18,  9,  3, 40, 15,  1, 90,  2,  5, 41, 12, 52,
75, 35],
       device='cuda:0')
        TOTAL (correct/total): 792.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.4102, Validation Accuracy: 91.6667 %
Epoch 7, Train Loss: 0.0043, Train Accuracy: 99.8335 %
         ---------------------------------------
        Predicted : tensor([58, 92,  0,  1,  3, 82, 64, 95,  5, 32,  3,  1, 84,
83,  5,  3],
       device='cuda:0')
        Result : tensor([58, 92,  0,  1,  3, 14, 64, 95,  5, 23,  3,  1, 84, 83,
5,  3],
       device='cuda:0')
        TOTAL (correct/total): 795.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.3939, Validation Accuracy: 92.0139 %

 ---------------------------------------
```



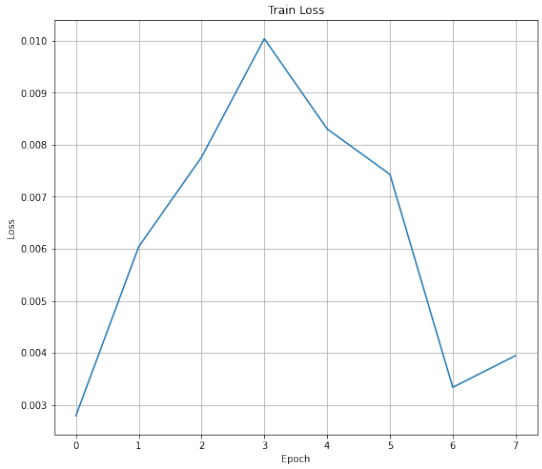


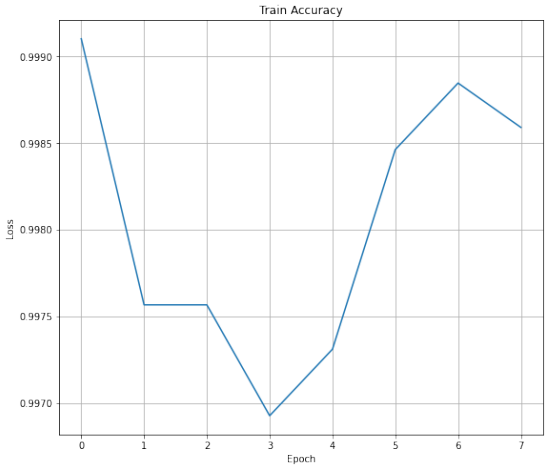


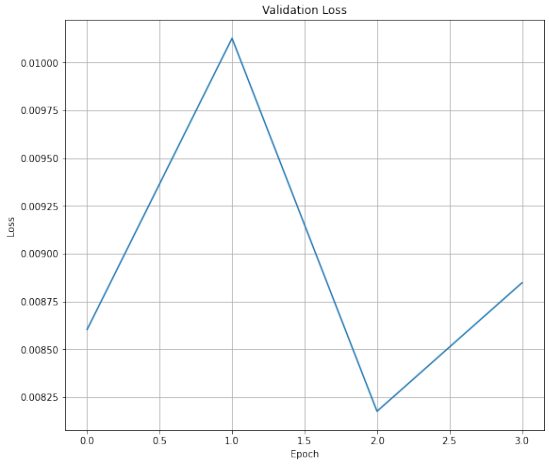


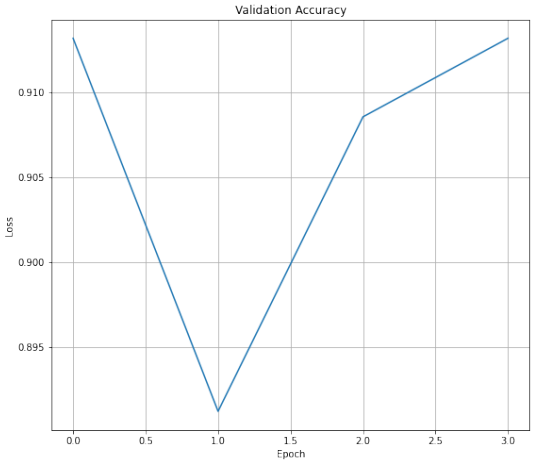

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 0:41:18 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0028, Train Accuracy: 99.9103 %
        ----------------------------------------
       Predicted : tensor([15,  3,  3, 29, 31, 41, 46,  1, 33, 61, 59,  9,  0,
15, 56, 68],
```

```
       device='cuda:0')
        Result : tensor([15,  3,  3, 26, 31, 41, 56,  1, 33, 61, 59,  9,  0, 15,
56, 68],
       device='cuda:0')
        TOTAL (correct/total): 789.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.4646, Validation Accuracy: 91.3194 %
Epoch 3, Train Loss: 0.0078, Train Accuracy: 99.7567 %
         ---------------------------------------
        Predicted : tensor([92, 17, 15,  0, 92,  0, 64, 87,  3, 85,  3, 16, 69,
87, 24,  5],
       device='cuda:0')
        Result : tensor([92, 17, 15,  0, 92,  0, 64, 87,  3, 85,  3, 16, 69, 87,
25,  5],
       device='cuda:0')
        TOTAL (correct/total): 770.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.5469, Validation Accuracy: 89.1204 %
Epoch 5, Train Loss: 0.0083, Train Accuracy: 99.7310 %
         ---------------------------------------
        Predicted : tensor([ 3, 69, 35, 81,  3,  3, 47, 15, 99, 45,  5, 90, 77,
23, 22,  5],
       device='cuda:0')
        Result : tensor([ 3, 69, 35, 81,  3,  3, 47, 15, 78, 45,  5, 90, 77, 79,
87,  5],
       device='cuda:0')
        TOTAL (correct/total): 785.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.4415, Validation Accuracy: 90.8565 %
Epoch 7, Train Loss: 0.0033, Train Accuracy: 99.8847 %
         ---------------------------------------
        Predicted : tensor([ 0, 98,  5, 92,  0, 72, 19, 86,  5,  3,  3, 82,  5,
32, 20, 88],
       device='cuda:0')
        Result : tensor([ 0, 98,  5, 92,  0, 72, 19, 86,  5,  3,  3, 82,  5, 32,
20, 88],
       device='cuda:0')
        TOTAL (correct/total): 789.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.4778, Validation Accuracy: 91.3194 %

 ---------------------------------------
```



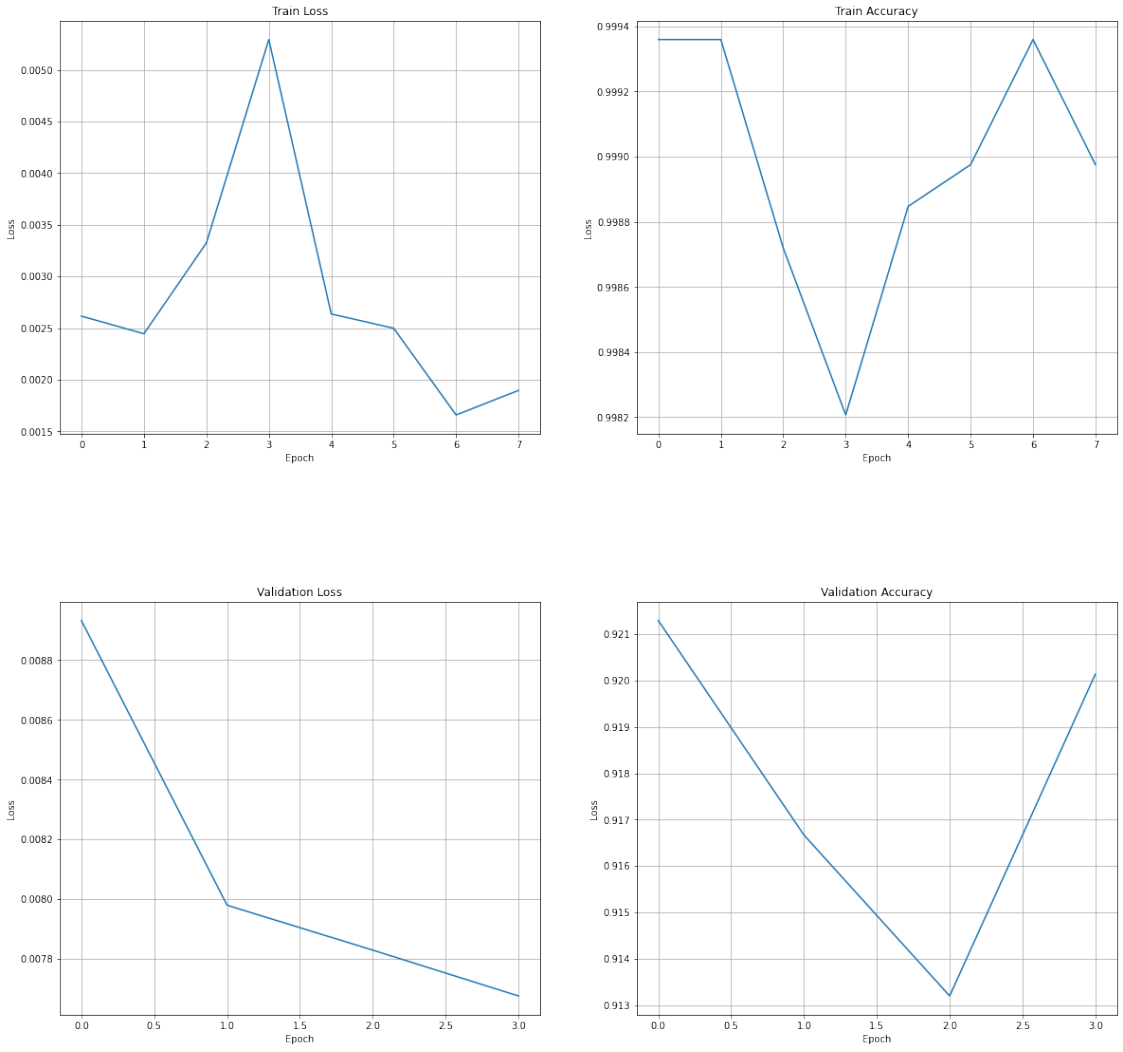







```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 0:46:36 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0026, Train Accuracy: 99.9360 %
        ----------------------------------------
       Predicted : tensor([54,  5,  3, 41, 15,  0,  0, 76,  0, 91, 23, 55, 36,
81, 27, 86],
```

```
       device='cuda:0')
        Result : tensor([54,  5,  3, 41, 15,  0,  0, 76,  0, 91, 23, 55, 18, 81,
27, 86],
       device='cuda:0')
        TOTAL (correct/total): 796.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.4824, Validation Accuracy: 92.1296 %
Epoch 3, Train Loss: 0.0033, Train Accuracy: 99.8719 %
         ---------------------------------------
        Predicted : tensor([ 5, 46,  0, 30,  0, 55, 97, 16, 62, 76, 63, 76,  5,
49, 69, 94],
       device='cuda:0')
        Result : tensor([ 5, 46,  0, 30,  0, 55, 97, 16, 62, 76, 63, 76,  5, 49,
69, 94],
       device='cuda:0')
        TOTAL (correct/total): 792.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.4309, Validation Accuracy: 91.6667 %
Epoch 5, Train Loss: 0.0026, Train Accuracy: 99.8847 %
         ---------------------------------------
        Predicted : tensor([42,  3, 75, 57, 99, 86,  5,  3,  1,  5, 10, 12, 83,
94,  0,  1],
       device='cuda:0')
        Result : tensor([42,  3, 75, 45,  6, 86,  5,  3,  1,  5, 10, 12, 83, 94,
0,  1],
       device='cuda:0')
        TOTAL (correct/total): 789.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.4228, Validation Accuracy: 91.3194 %
Epoch 7, Train Loss: 0.0017, Train Accuracy: 99.9360 %
         ---------------------------------------
        Predicted : tensor([ 0, 55, 24,  3, 50,  3,  5, 19,  3,  0, 56, 12,  0,
2,  3, 85],
       device='cuda:0')
        Result : tensor([ 0, 55, 24,  3, 50,  3,  5, 19,  3,  0, 56, 12,  0,  2,
3, 85],
       device='cuda:0')
        TOTAL (correct/total): 795.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.4145, Validation Accuracy: 92.0139 %

 ---------------------------------------
```




```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 2844), started 0:51:52 ago. (Use '!kill␣
 ↪2844' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

```
La moyenne des accuracy: 362.2917 %
```

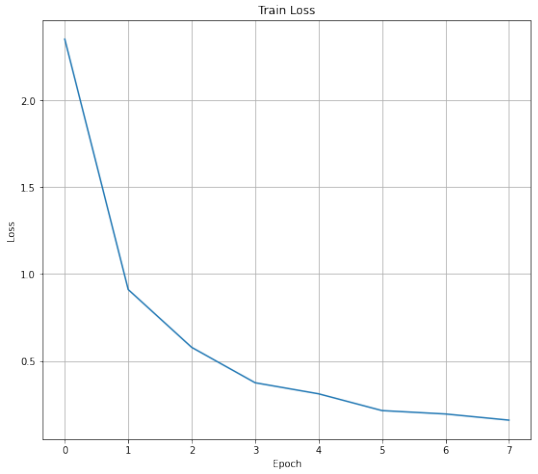
6.3 Pour le modèle squeezenet1_0 avec K = 5

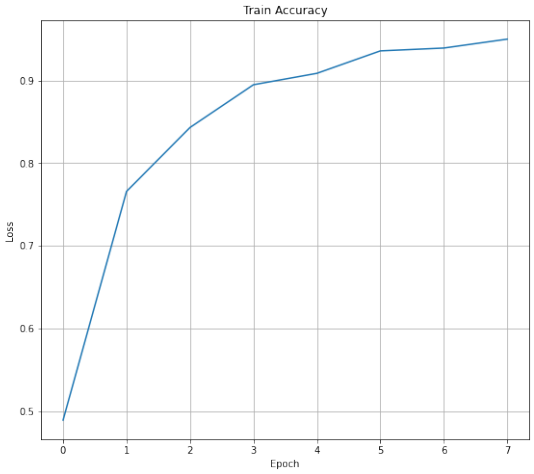
[ ]:
```
model_squeezenet1_0 = adapt_pretrained_model("squeezenet1_0")
loss_fn = nn.CrossEntropyLoss()
optimiseur_squeezenet1_0 = torch.optim.SGD(model_squeezenet1_0.parameters(),␣
 ↪lr=0.001, momentum=0.9)
```

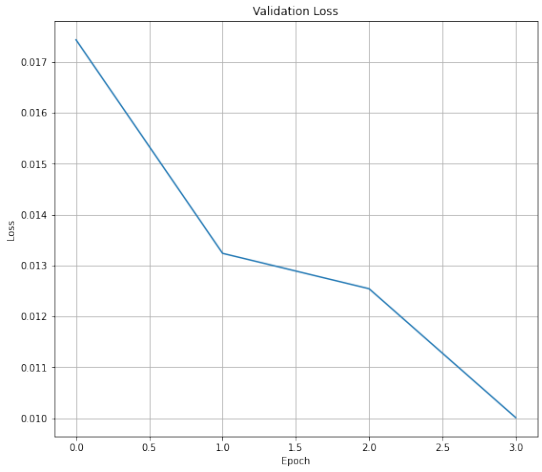
```
mean_squeezenet1_0 =␣
 ↪boucle_de_validation_croisee_pour_k_iteration(model_squeezenet1_0,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_squeezenet1_0,␣
 ↪epochs, typeTrain="squeezenet1_0")
```

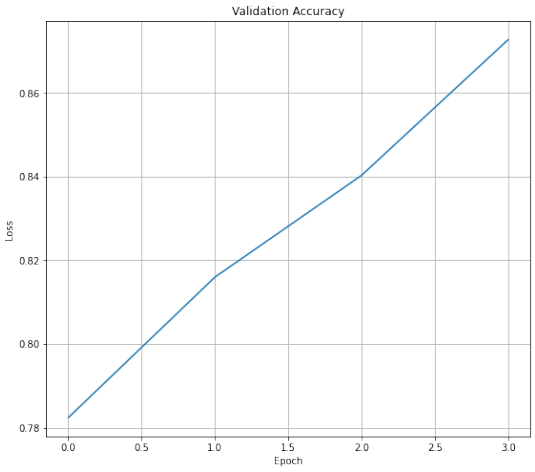
```
Downloading: "https://download.pytorch.org/models/squeezenet1_0-b66bff10.pth" to
/root/.cache/torch/hub/checkpoints/squeezenet1_0-b66bff10.pth

  0%|          | 0.00/4.78M [00:00<?, ?B/s]

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 2.3528, Train Accuracy: 48.8986 %
         ----------------------------------------
        Predicted : tensor([69, 49, 26,  0, 14, 82, 77, 77,  5,  3,  2, 93, 16,
5,  5,  3],
       device='cuda:0')
        Result : tensor([69, 54, 72,  0, 14, 82, 77, 77,  5,  3,  2, 93, 16,  5,
5,  3],
       device='cuda:0')
        TOTAL (correct/total): 676.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.9414, Validation Accuracy: 78.2407 %
Epoch 3, Train Loss: 0.5781, Train Accuracy: 84.3110 %
         ----------------------------------------
        Predicted : tensor([52, 59, 86, 51, 90, 94, 51,  2, 81, 41,  3, 84, 10,
61, 92,  0],
       device='cuda:0')
        Result : tensor([52, 27, 87, 51, 90, 94, 76,  2, 44, 41,  3, 84, 10, 61,
92,  0],
       device='cuda:0')
        TOTAL (correct/total): 705.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.7149, Validation Accuracy: 81.5972 %
Epoch 5, Train Loss: 0.3107, Train Accuracy: 90.8683 %
         ----------------------------------------
        Predicted : tensor([ 90,  27,  13,  68,  35,   3,  51,   3,  15,  94,
81,  94,  16,  72,
         36, 100], device='cuda:0')
        Result : tensor([ 90,  26,  12,  68,  23,   3,  51,   3,  15,  94,  81,
94,  16,  72,
         10, 100], device='cuda:0')
        TOTAL (correct/total): 726.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.6773, Validation Accuracy: 84.0278 %
```

```
Epoch 7, Train Loss: 0.1945, Train Accuracy: 93.9293 %
         ----------------------------------------
        Predicted : tensor([94, 90,  2, 19, 19,  0, 72,  0,  3,  5, 94, 65, 45,
46, 94,  3],
       device='cuda:0')
        Result : tensor([94, 90,  2, 19, 19,  0, 72,  0,  3,  5, 94, 65, 45, 46,
94,  3],
       device='cuda:0')
        TOTAL (correct/total): 754.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.5407, Validation Accuracy: 87.2685 %

 ----------------------------------------
```










```
The tensorboard extension is already loaded. To reload it, use:
```

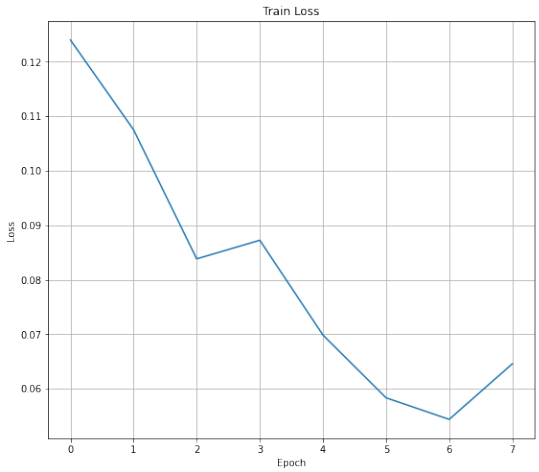
```
  %reload_ext tensorboard
```

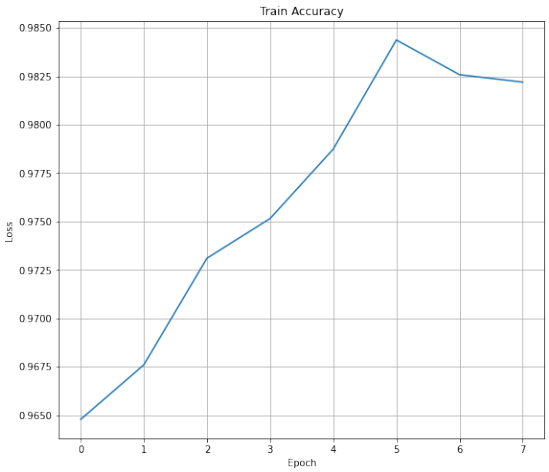
```
Reusing TensorBoard on port 6006 (pid 2844), started 0:58:04 ago. (Use '!kill␣
 ↪2844' to kill it.)
```

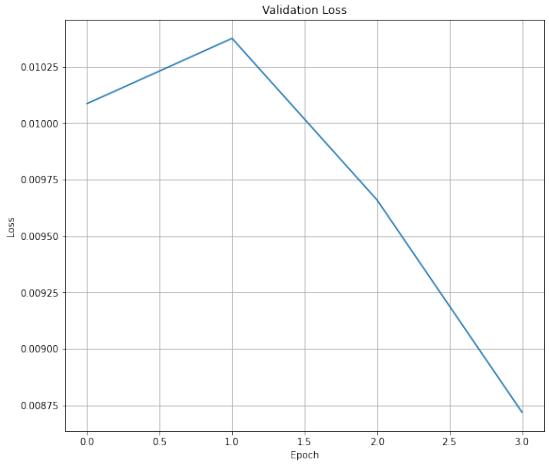
```
<IPython.core.display.Javascript object>
```

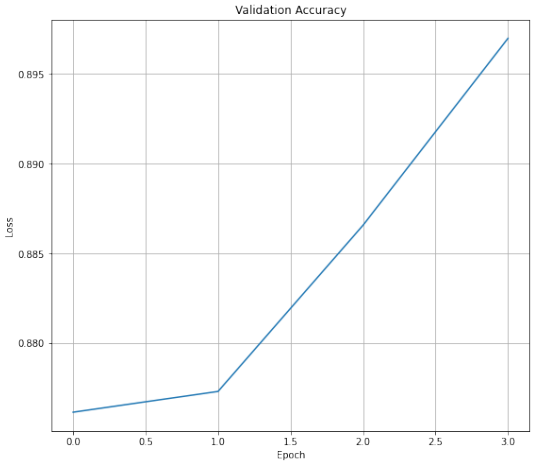
```
La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.1240, Train Accuracy: 96.4780 %
         ----------------------------------------
        Predicted : tensor([ 3, 39,  3,  1, 23, 72, 12, 79,  5,  0,  5, 49,  1,
45,  3,  3],
       device='cuda:0')
        Result : tensor([ 3, 39,  3,  1, 23, 72, 12, 79,  5,  0,  5, 49,  1, 45,
3,  3],
       device='cuda:0')
        TOTAL (correct/total): 757.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.5447, Validation Accuracy: 87.6157 %
Epoch 3, Train Loss: 0.0838, Train Accuracy: 97.3105 %
         ----------------------------------------
        Predicted : tensor([50, 95, 41,  5, 27,  5, 22,  0, 27, 12,  3, 37,  5,
69,  2, 58],
       device='cuda:0')
        Result : tensor([50, 95, 41,  5, 29,  5, 22, 25, 27, 12,  3, 37,  5, 69,
2, 54],
       device='cuda:0')
        TOTAL (correct/total): 758.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.5603, Validation Accuracy: 87.7315 %
Epoch 5, Train Loss: 0.0699, Train Accuracy: 97.8740 %
         ----------------------------------------
        Predicted : tensor([96,  3, 23, 34, 26,  5,  2, 50, 12, 88, 64, 92, 94,
58, 69, 47],
       device='cuda:0')
        Result : tensor([96,  3, 23, 34, 81,  5,  2, 50, 12, 88, 64, 92, 94, 51,
69, 47],
       device='cuda:0')
        TOTAL (correct/total): 766.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.5216, Validation Accuracy: 88.6574 %
Epoch 7, Train Loss: 0.0544, Train Accuracy: 98.2582 %
         ----------------------------------------
        Predicted : tensor([88, 72, 73,  1,  3,  3, 41, 36,  1, 93, 71,  3, 58,
3, 40, 77],
```

```
       device='cuda:0')
        Result : tensor([88, 72, 73,  1,  3,  3, 41, 36,  1, 93, 71,  3, 58,  3,
40, 77],
       device='cuda:0')
        TOTAL (correct/total): 775.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.4708, Validation Accuracy: 89.6991 %

 ----------------------------------------
```





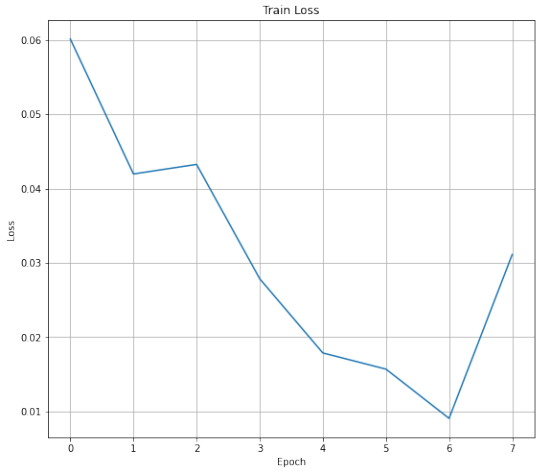


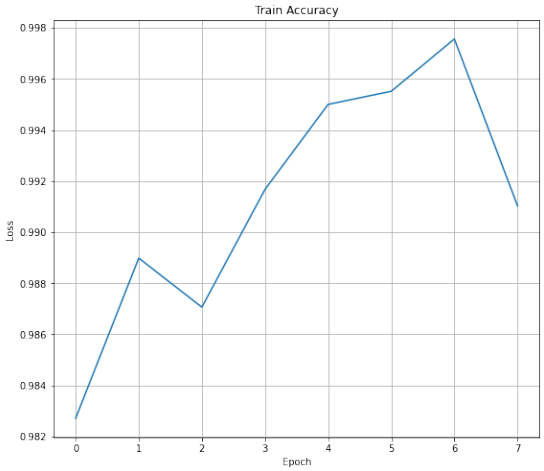


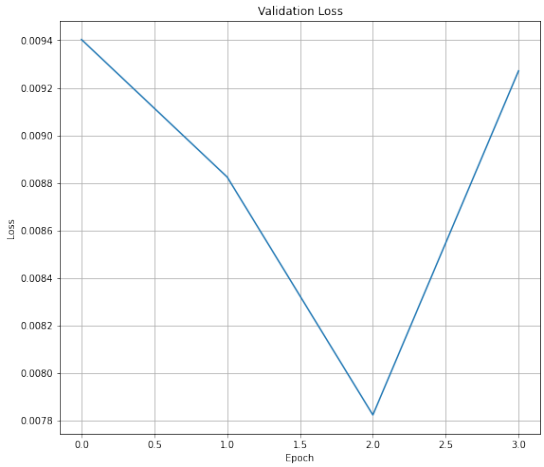

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 1:04:14 ago. (Use '!kill␣
 ↪2844' to kill it.)
```

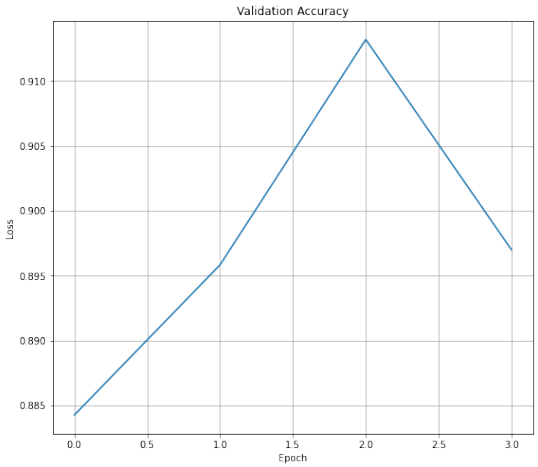
```
<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0601, Train Accuracy: 98.2710 %
         ----------------------------------------
        Predicted : tensor([97,  3, 22,  5, 97,  3, 60, 25,  3,  2, 16, 90, 30,
59,  3, 29],
       device='cuda:0')
        Result : tensor([97,  3, 22,  5, 97,  3, 60, 25,  3,  2, 16, 90, 30, 59,
3, 29],
       device='cuda:0')
        TOTAL (correct/total): 764.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.5078, Validation Accuracy: 88.4259 %
Epoch 3, Train Loss: 0.0432, Train Accuracy: 98.7065 %
         ----------------------------------------
        Predicted : tensor([ 3,  3, 60, 94, 80, 98,  5,  3, 85, 51,  0, 53, 54,
95, 84, 40],
       device='cuda:0')
        Result : tensor([ 3,  3, 60, 94, 80, 98,  5,  3, 85, 51,  0, 58, 54, 95,
84, 40],
       device='cuda:0')
        TOTAL (correct/total): 774.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.4765, Validation Accuracy: 89.5833 %
Epoch 5, Train Loss: 0.0179, Train Accuracy: 99.5005 %
         ----------------------------------------
        Predicted : tensor([47,  8,  3, 12, 19, 25, 83, 54, 31, 63, 41,  0,  1,
3, 34, 90],
       device='cuda:0')
        Result : tensor([47,  8,  3, 12, 19, 25, 83, 54, 31, 63, 41,  0,  1,  3,
34, 90],
       device='cuda:0')
        TOTAL (correct/total): 789.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.4225, Validation Accuracy: 91.3194 %
Epoch 7, Train Loss: 0.0091, Train Accuracy: 99.7567 %
         ----------------------------------------
        Predicted : tensor([54, 25,  3, 29,  2, 50, 12, 12,  5, 50,  0, 16, 98,
19, 55, 85],
       device='cuda:0')
        Result : tensor([54, 25,  3, 26,  2, 50, 12, 12,  5, 50,  0, 16, 98, 19,
55, 85],
       device='cuda:0')
```

```
        TOTAL (correct/total): 775.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.5007, Validation Accuracy: 89.6991 %

----------------------------------------
```



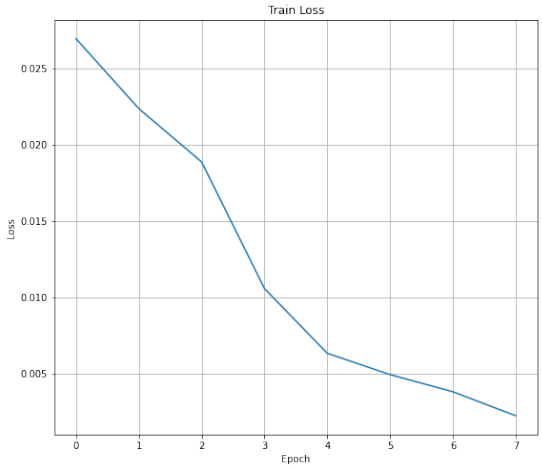


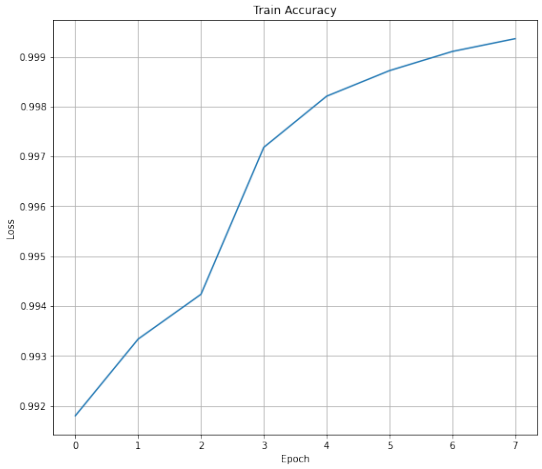


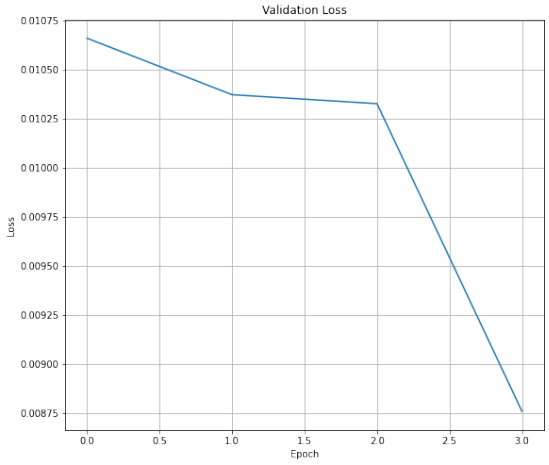


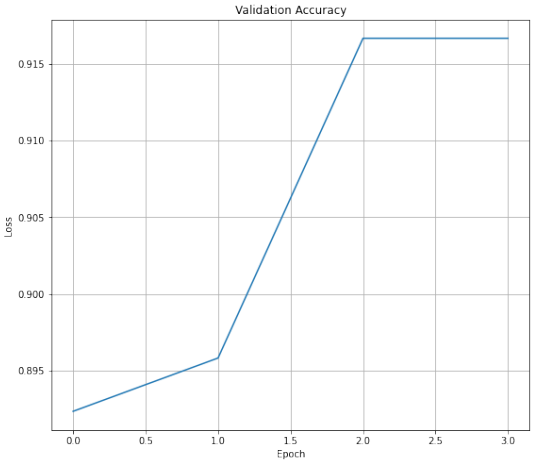

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 1:10:21 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0
```

```
 ---------------------------------------
Epoch 1, Train Loss: 0.0269, Train Accuracy: 99.1803 %
         ---------------------------------------
        Predicted : tensor([24, 23, 57, 23, 12,  0,  2, 75, 42, 36, 62, 44, 84,
57,  1, 69],
       device='cuda:0')
        Result : tensor([36, 15, 57, 23, 12,  0,  2, 75, 42, 14, 62, 44, 84, 57,
1, 69],
       device='cuda:0')
        TOTAL (correct/total): 771.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.5755, Validation Accuracy: 89.2361 %
Epoch 3, Train Loss: 0.0189, Train Accuracy: 99.4237 %
         ---------------------------------------
        Predicted : tensor([ 5, 56, 75, 87,  5, 86, 13,  2,  5, 23, 98, 24, 19,
23, 40,  3],
       device='cuda:0')
        Result : tensor([ 5, 56, 75, 87,  5, 86, 13,  2,  5, 23, 98, 24, 19, 56,
40,  3],
       device='cuda:0')
        TOTAL (correct/total): 774.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.5600, Validation Accuracy: 89.5833 %
Epoch 5, Train Loss: 0.0063, Train Accuracy: 99.8207 %
         ---------------------------------------
        Predicted : tensor([ 5,  5, 16, 24,  3, 18,  3, 43, 40,  3, 94,  5,  0,
38, 59, 60],
       device='cuda:0')
        Result : tensor([ 5,  5, 16, 25,  3, 20,  3, 43, 40,  3, 94,  5,  0, 38,
27, 95],
       device='cuda:0')
        TOTAL (correct/total): 792.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.5575, Validation Accuracy: 91.6667 %
Epoch 7, Train Loss: 0.0038, Train Accuracy: 99.9103 %
         ---------------------------------------
        Predicted : tensor([ 12,  42,  16,  55,  83,  19,  90,  10,   5,  12,
100,  54,  50,  26,
          5,  98], device='cuda:0')
        Result : tensor([ 12,  39,  16,  55,  83,  19,  90,  10,   5,  12, 100,
54,  50,  26,
          5,  98], device='cuda:0')
        TOTAL (correct/total): 792.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.4729, Validation Accuracy: 91.6667 %

 ---------------------------------------
```



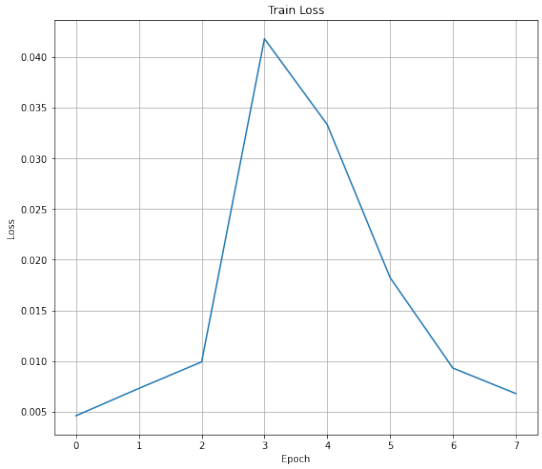


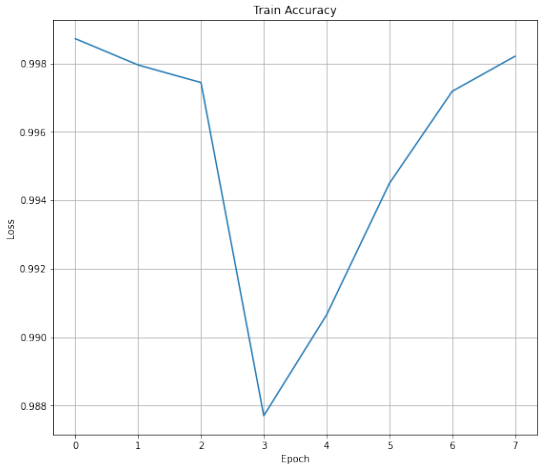


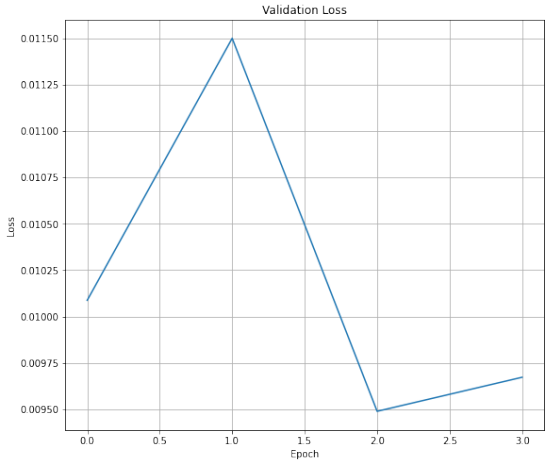


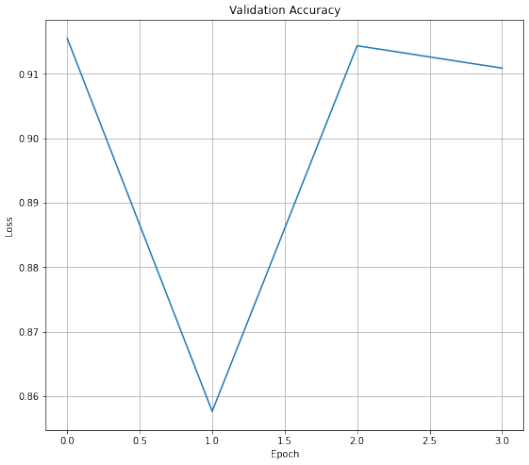

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 1:16:26 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0046, Train Accuracy: 99.8719 %
        ----------------------------------------
       Predicted : tensor([94, 46,  3, 97,  3, 92, 15,  3, 32, 36, 39, 77,  5,
86,  1, 96],
```

```
       device='cuda:0')
        Result : tensor([94, 46,  3, 97,  3, 92, 15,  3, 32, 36, 39, 77,  5, 86,
1, 96],
       device='cuda:0')
        TOTAL (correct/total): 791.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.5447, Validation Accuracy: 91.5509 %
Epoch 3, Train Loss: 0.0099, Train Accuracy: 99.7439 %
         ---------------------------------------
        Predicted : tensor([19, 13, 94, 54, 23,  5,  6, 93, 41, 75,  3, 94, 64,
15, 49, 31],
       device='cuda:0')
        Result : tensor([19, 13, 94, 44, 23, 50,  6, 93, 41, 75,  3, 94, 64, 52,
49, 31],
       device='cuda:0')
        TOTAL (correct/total): 741.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.6210, Validation Accuracy: 85.7639 %
Epoch 5, Train Loss: 0.0333, Train Accuracy: 99.0651 %
         ---------------------------------------
        Predicted : tensor([ 5,  3,  5,  5,  3,  5, 34,  4, 86, 87,  2,  2,  3,
40,  5, 57],
       device='cuda:0')
        Result : tensor([ 5,  3,  5,  5,  3,  5, 34,  4, 86, 87,  2,  2,  3, 40,
5, 57],
       device='cuda:0')
        TOTAL (correct/total): 790.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.5124, Validation Accuracy: 91.4352 %
Epoch 7, Train Loss: 0.0093, Train Accuracy: 99.7182 %
         ---------------------------------------
        Predicted : tensor([ 1, 93, 31,  3,  3, 66, 83,  5, 35, 94, 55, 95, 54,
90,  9,  3],
       device='cuda:0')
        Result : tensor([ 1, 93, 31,  3,  3, 49, 83,  5, 35, 94, 55, 95, 54, 90,
9,  3],
       device='cuda:0')
        TOTAL (correct/total): 787.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.5223, Validation Accuracy: 91.0880 %

 ---------------------------------------
```










```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 2844), started 1:22:29 ago. (Use '!kill␣
 ↪2844' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

```
La moyenne des accuracy: 353.1713 %
```

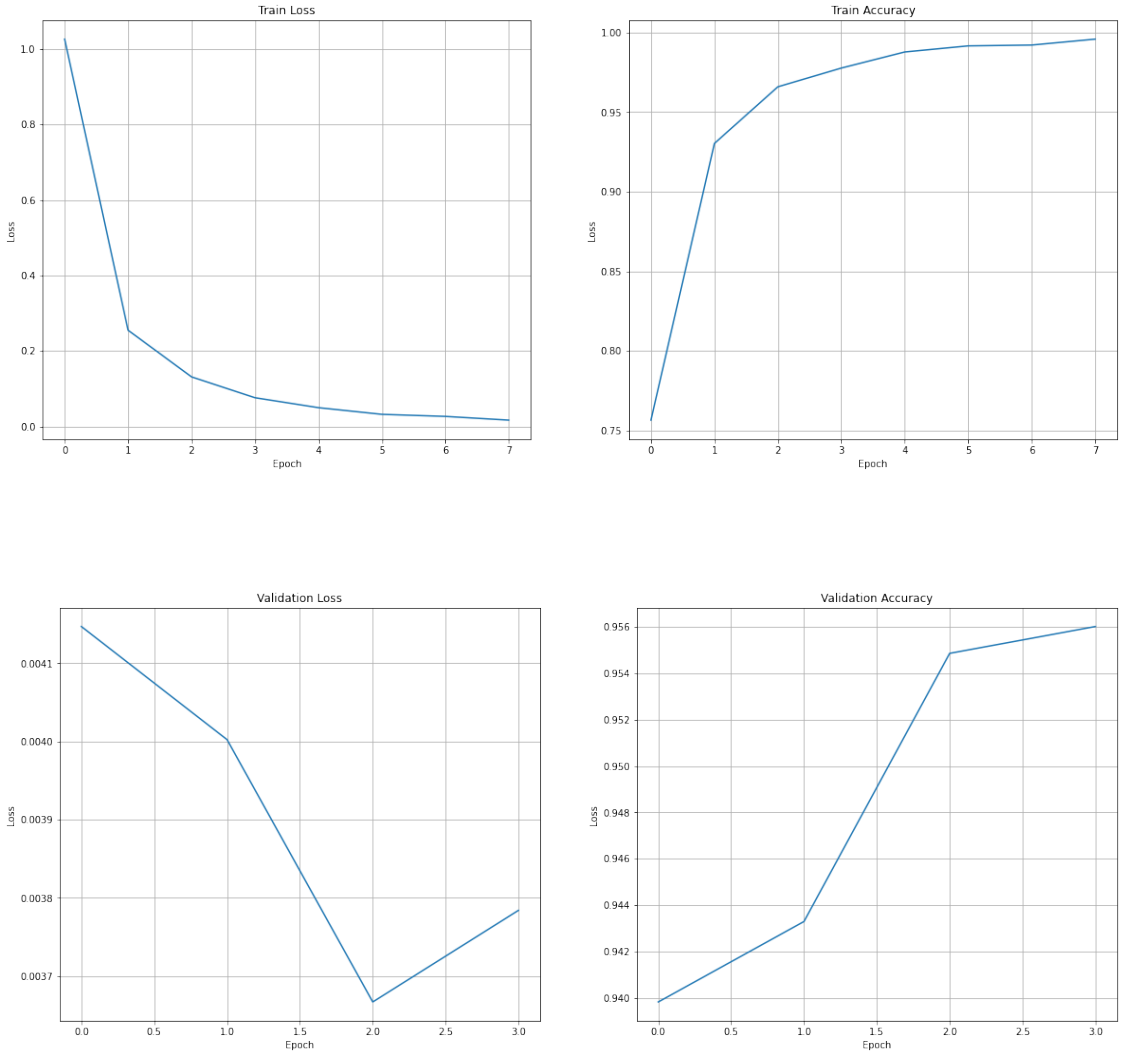
6.4 Pour le modèle vgg16 avec K = 5

[ ]:
```
model_vgg16 = adapt_pretrained_model("vgg16")
loss_fn = nn.CrossEntropyLoss()
optimiseur_vgg16 = torch.optim.SGD(model_vgg16.parameters(), lr=0.001,␣
 ↪momentum=0.9)
```

```
mean_vgg16 = boucle_de_validation_croisee_pour_k_iteration(model_vgg16,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_vgg16, epochs,␣
 ↪typeTrain="vgg16")
```

```
Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to
/root/.cache/torch/hub/checkpoints/vgg16-397923af.pth

  0%|          | 0.00/528M [00:00<?, ?B/s]

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 1.0260, Train Accuracy: 75.6404 %
         ----------------------------------------
        Predicted : tensor([46, 82,  3, 18, 90, 45, 14, 25, 31, 94, 40,  2,  1,
3,  3,  2],
       device='cuda:0')
        Result : tensor([46, 67,  3, 18, 90, 45, 14, 27, 31, 18, 40,  2,  1,  3,
3,  2],
       device='cuda:0')
        TOTAL (correct/total): 812.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.2239, Validation Accuracy: 93.9815 %
Epoch 3, Train Loss: 0.1311, Train Accuracy: 96.5932 %
         ----------------------------------------
        Predicted : tensor([16, 89, 69, 42,  3,  3, 75,  5,  5, 47,  1, 44, 18,
62, 66,  0],
       device='cuda:0')
        Result : tensor([16, 89, 56, 42,  3,  3, 75,  5,  5, 47,  1, 44, 18, 62,
66,  0],
       device='cuda:0')
        TOTAL (correct/total): 815.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.2161, Validation Accuracy: 94.3287 %
Epoch 5, Train Loss: 0.0496, Train Accuracy: 98.7833 %
         ----------------------------------------
        Predicted : tensor([ 2,  3, 29, 83,  1, 96, 60, 94,  3,  1, 23, 55, 25,
87, 44, 94],
       device='cuda:0')
        Result : tensor([ 2,  3, 29, 83,  1, 96, 60, 94,  3,  1, 15, 55, 27, 87,
44, 18],
       device='cuda:0')
        TOTAL (correct/total): 825.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.1980, Validation Accuracy: 95.4861 %
Epoch 7, Train Loss: 0.0265, Train Accuracy: 99.2188 %
```

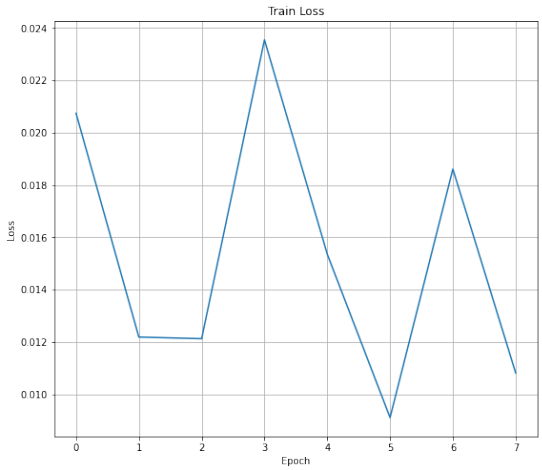
```
 ----------------------------------------
Predicted : tensor([12, 19,  9,  2, 20, 16, 33,  5,  3,  0,  2,  5,  1,
48, 90, 30],
      device='cuda:0')
 Result : tensor([12, 19,  9,  2, 20, 16, 56,  5,  3,  0,  2,  5,  1, 83,
90, 30],
      device='cuda:0')
 TOTAL (correct/total): 826.0 / 864.0
 ----------------------------------------
 Epoch 8, Validation Loss: 0.2043, Validation Accuracy: 95.6019 %

----------------------------------------
```

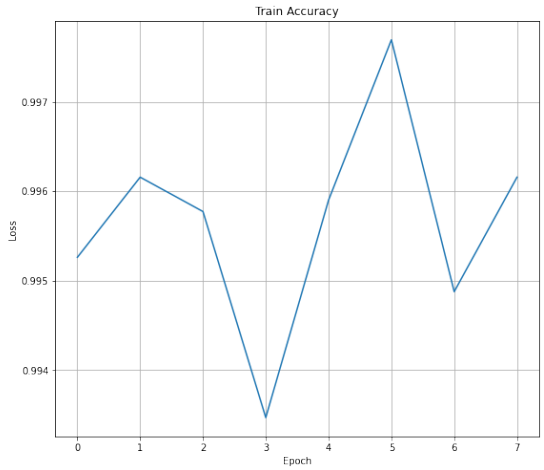


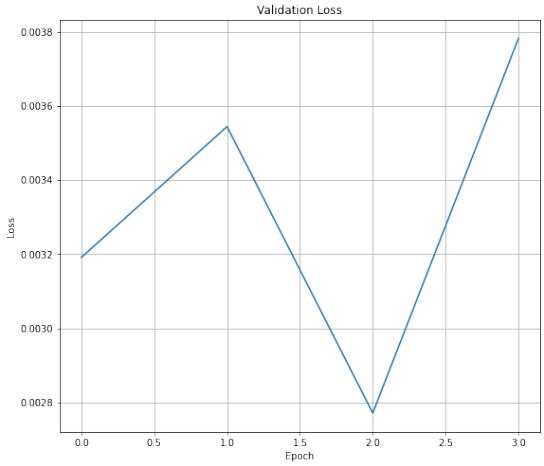

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

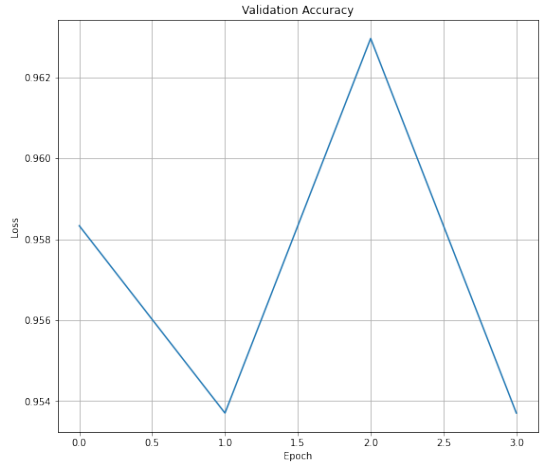
```
Reusing TensorBoard on port 6006 (pid 2844), started 1:42:15 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0207, Train Accuracy: 99.5261 %
         ----------------------------------------
        Predicted : tensor([ 3,  3,  5,  2, 86, 39, 19, 94, 53, 88, 47, 37, 51,
3,  5,  3],
       device='cuda:0')
        Result : tensor([ 3,  3,  5,  2, 86, 39, 19, 94, 53, 88, 47, 37, 51,  3,
5,  3],
       device='cuda:0')
        TOTAL (correct/total): 828.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.1723, Validation Accuracy: 95.8333 %
Epoch 3, Train Loss: 0.0121, Train Accuracy: 99.5774 %
         ----------------------------------------
        Predicted : tensor([ 1, 26,  3,  5, 49, 26, 44,  5, 62, 81, 13, 23, 12,
1, 15,  3],
       device='cuda:0')
        Result : tensor([ 1, 26,  3,  5, 49, 26, 44,  5, 62, 81, 13, 23, 12,  1,
15,  3],
       device='cuda:0')
        TOTAL (correct/total): 824.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.1914, Validation Accuracy: 95.3704 %
Epoch 5, Train Loss: 0.0154, Train Accuracy: 99.5902 %
         ----------------------------------------
        Predicted : tensor([25,  5, 23, 56, 51, 54, 90,  3, 41,  5, 71, 30, 68,
32, 27, 35],
       device='cuda:0')
        Result : tensor([25,  5, 23, 56, 51, 54, 90,  3, 41,  5, 71, 30, 44, 32,
27, 35],
       device='cuda:0')
        TOTAL (correct/total): 832.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.1497, Validation Accuracy: 96.2963 %
Epoch 7, Train Loss: 0.0186, Train Accuracy: 99.4877 %
         ----------------------------------------
        Predicted : tensor([96, 90, 32, 56, 62, 23, 72,  3,  1, 51, 86,  1,  3,
62,  2, 90],
       device='cuda:0')
```

```
        Result : tensor([96, 90, 32, 56, 62, 23, 72,  3,  1, 51, 86,  1,  3, 62,
2, 90],
       device='cuda:0')
        TOTAL (correct/total): 824.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.2042, Validation Accuracy: 95.3704 %

 ---------------------------------------
```









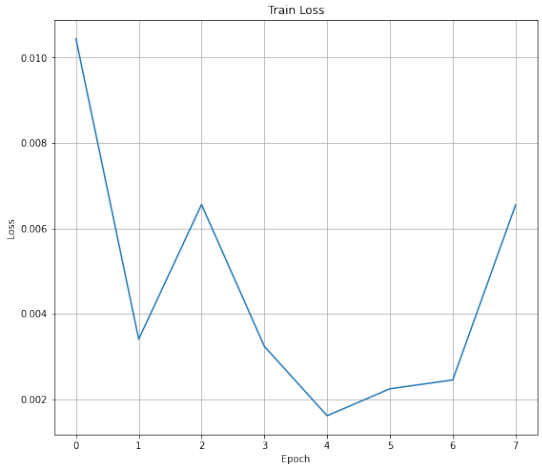

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

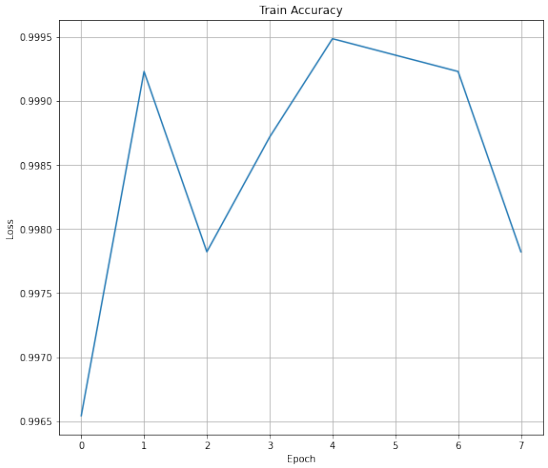
```
Reusing TensorBoard on port 6006 (pid 2844), started 2:02:00 ago. (Use '!kill␣
↪2844' to kill it.)
```

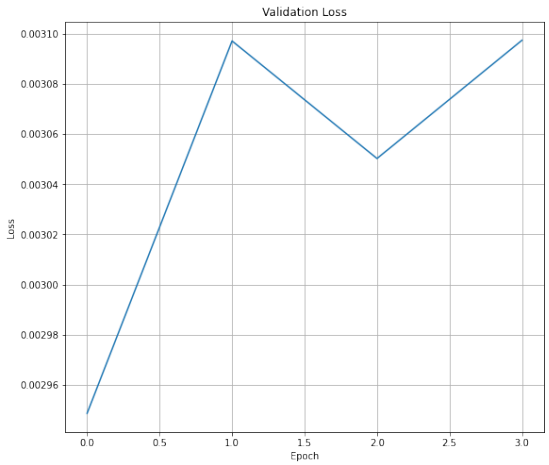
```
<IPython.core.display.Javascript object>
```

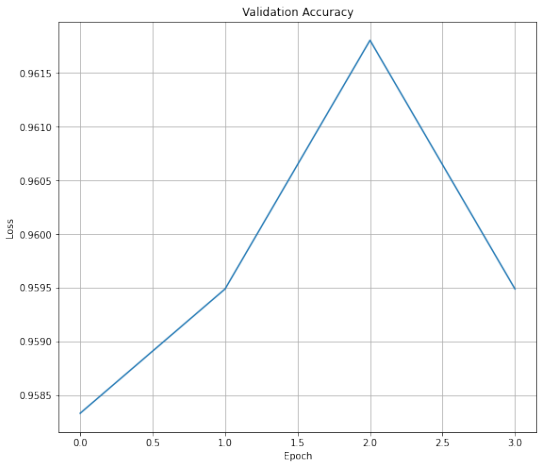
```
La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0104, Train Accuracy: 99.6542 %
         ----------------------------------------
        Predicted : tensor([44, 61, 56, 12, 41,  3, 64,  3,  0, 15, 69, 18, 76,
40,  0,  0],
       device='cuda:0')
        Result : tensor([44, 61, 56, 12, 41,  3, 64,  3,  0, 15, 69, 18, 76, 40,
0,  0],
       device='cuda:0')
        TOTAL (correct/total): 828.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.1592, Validation Accuracy: 95.8333 %
Epoch 3, Train Loss: 0.0066, Train Accuracy: 99.7823 %
         ----------------------------------------
        Predicted : tensor([ 3,  3, 26, 90, 76,  0, 85, 34, 34, 94, 34, 52,  3,
25, 15, 44],
       device='cuda:0')
        Result : tensor([ 3,  3, 26, 90, 76,  0, 85, 34, 67, 94, 34, 52,  3, 25,
15, 44],
       device='cuda:0')
        TOTAL (correct/total): 829.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.1672, Validation Accuracy: 95.9491 %
Epoch 5, Train Loss: 0.0016, Train Accuracy: 99.9488 %
         ----------------------------------------
        Predicted : tensor([35, 12, 41,  3, 81, 23, 68, 39,  5, 93,  3,  4, 31,
39, 75, 90],
       device='cuda:0')
        Result : tensor([35, 12, 41,  3, 81, 23, 68, 39,  5, 93,  3,  4, 31, 39,
75, 90],
       device='cuda:0')
        TOTAL (correct/total): 831.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.1647, Validation Accuracy: 96.1806 %
Epoch 7, Train Loss: 0.0025, Train Accuracy: 99.9232 %
         ----------------------------------------
        Predicted : tensor([36, 19, 49,  3,  3, 19, 12, 64,  3, 57,  5,  2, 30,
71, 68, 26],
       device='cuda:0')
        Result : tensor([36, 19, 49,  3,  3, 19, 12, 64,  3, 57,  5,  2, 30, 71,
68, 81],
       device='cuda:0')
        TOTAL (correct/total): 829.0 / 864.0
         ----------------------------------------
```

```
Epoch 8, Validation Loss: 0.1673, Validation Accuracy: 95.9491 %

----------------------------------------
```



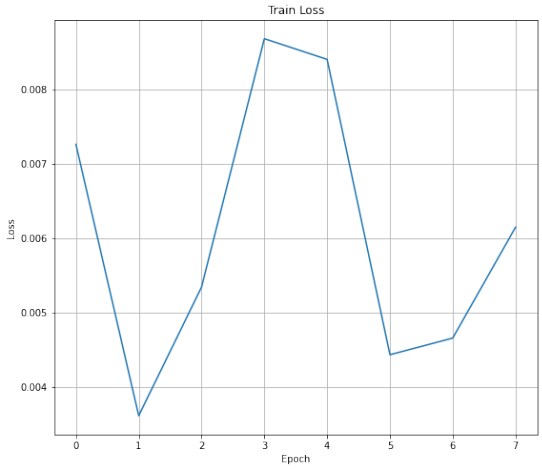


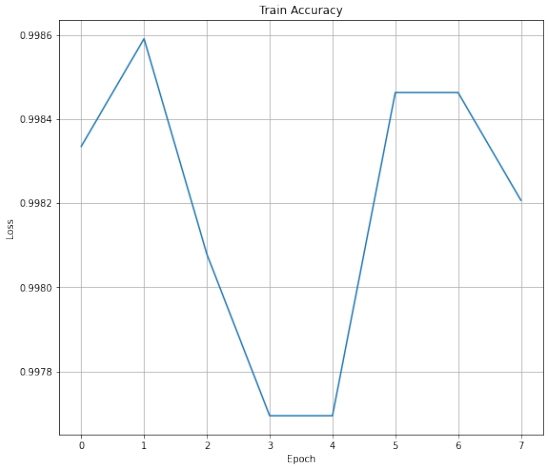


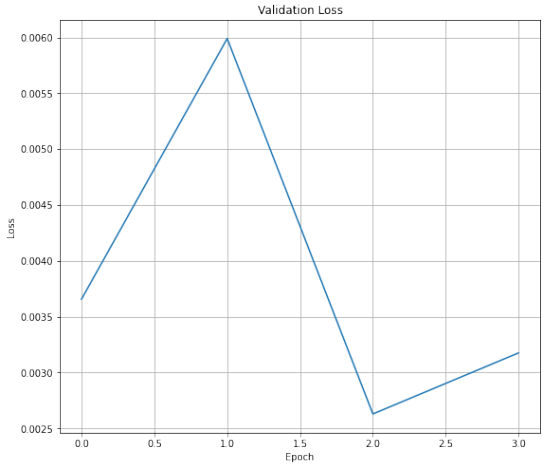


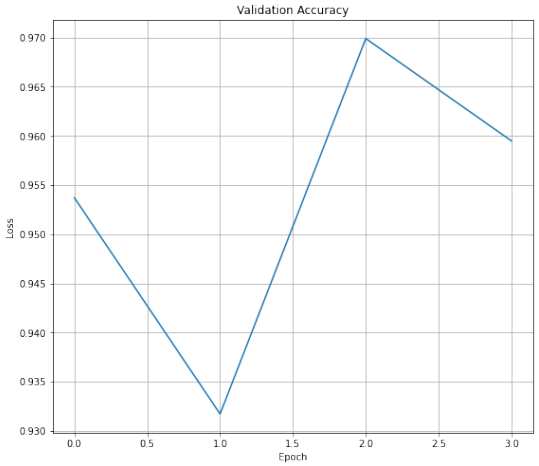

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 2:21:46 ago. (Use '!kill␣
  ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

----------------------------------------
```

```
Epoch 1, Train Loss: 0.0073, Train Accuracy: 99.8335 %
         ---------------------------------------
        Predicted : tensor([34,  5, 94,  2, 50, 27, 46,  1, 41, 68, 55, 71,  0,
5,  2, 94],
       device='cuda:0')
        Result : tensor([67,  5, 94,  2, 50, 59, 46,  1, 41, 68, 55, 71,  0,  5,
2, 94],
       device='cuda:0')
        TOTAL (correct/total): 824.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.1975, Validation Accuracy: 95.3704 %
Epoch 3, Train Loss: 0.0053, Train Accuracy: 99.8079 %
         ---------------------------------------
        Predicted : tensor([80,  3, 69, 23,  3, 33,  3, 39, 12, 49, 90, 94,  3,
91, 40, 50],
       device='cuda:0')
        Result : tensor([80,  3, 69, 23,  3, 33,  3, 39, 12, 49, 90, 94,  3, 91,
40, 50],
       device='cuda:0')
        TOTAL (correct/total): 805.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.3235, Validation Accuracy: 93.1713 %
Epoch 5, Train Loss: 0.0084, Train Accuracy: 99.7695 %
         ---------------------------------------
        Predicted : tensor([83, 36,  2, 41,  3, 92, 19, 32, 16, 15, 13, 72, 37,
3,  0, 42],
       device='cuda:0')
        Result : tensor([ 7, 36,  2, 41,  3, 92, 19, 32, 16, 15, 13, 72, 37,  3,
0, 42],
       device='cuda:0')
        TOTAL (correct/total): 838.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.1420, Validation Accuracy: 96.9907 %
Epoch 7, Train Loss: 0.0047, Train Accuracy: 99.8463 %
         ---------------------------------------
        Predicted : tensor([12, 94, 75,  2,  3, 73, 72,  5, 15, 57,  5, 27,  5,
79, 49, 77],
       device='cuda:0')
        Result : tensor([12, 94, 75,  2,  3, 73, 72,  5, 15, 57,  5, 27,  5, 79,
49, 77],
       device='cuda:0')
        TOTAL (correct/total): 829.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.1715, Validation Accuracy: 95.9491 %

 ---------------------------------------
```



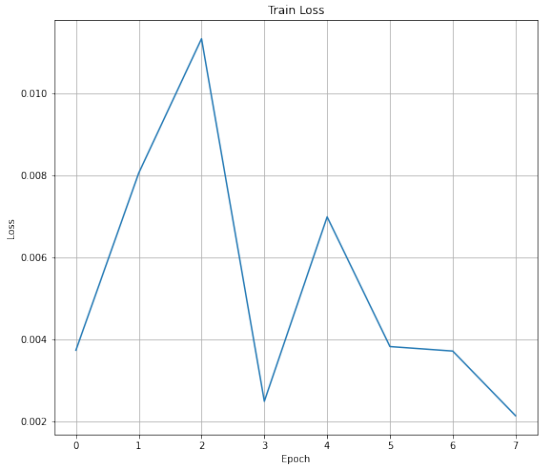


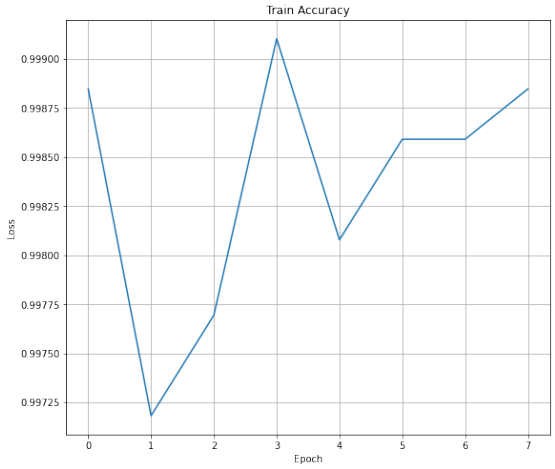


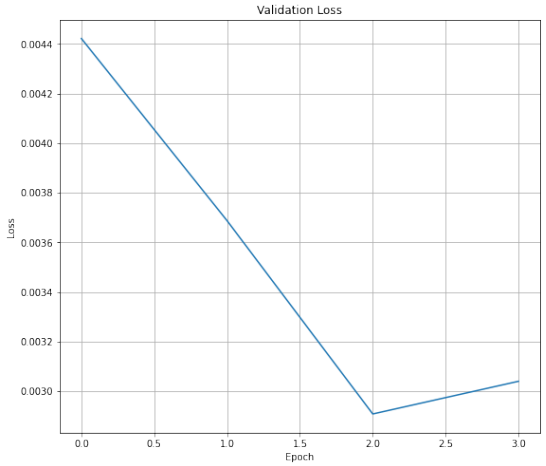


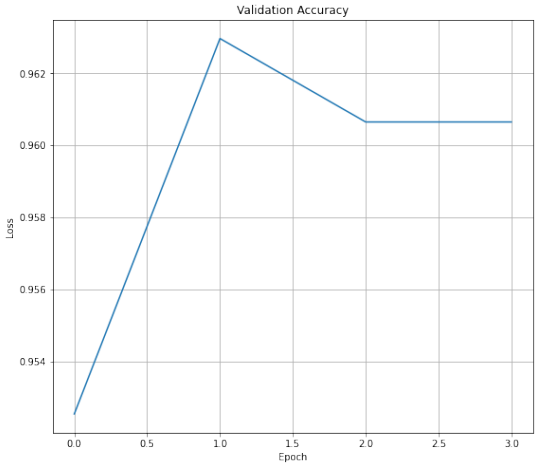

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 2844), started 2:41:31 ago. (Use '!kill␣
 ↪2844' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0037, Train Accuracy: 99.8847 %
        ----------------------------------------
       Predicted : tensor([16, 44,  0, 24,  5, 16,  3,  3, 86,  5, 94, 84, 19,
76, 98,  5],
```

```
       device='cuda:0')
        Result : tensor([16, 44,  0, 36,  5, 16,  3,  3, 86,  5, 94, 84, 19, 76,
98,  5],
       device='cuda:0')
        TOTAL (correct/total): 823.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.2388, Validation Accuracy: 95.2546 %
Epoch 3, Train Loss: 0.0113, Train Accuracy: 99.7695 %
         ---------------------------------------
        Predicted : tensor([75,  1, 63,  5, 94, 16, 40, 30,  3, 79, 39, 76, 94,
5, 27,  3],
       device='cuda:0')
        Result : tensor([75,  1, 63,  5, 18, 16, 40, 30,  3, 79, 39, 76, 94,  5,
27,  3],
       device='cuda:0')
        TOTAL (correct/total): 832.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.1991, Validation Accuracy: 96.2963 %
Epoch 5, Train Loss: 0.0070, Train Accuracy: 99.8079 %
         ---------------------------------------
        Predicted : tensor([63,  5, 73,  0, 41, 19,  5, 98, 20,  0, 49, 23, 32,
3, 23, 45],
       device='cuda:0')
        Result : tensor([23,  5, 73,  0, 41, 19,  5, 98, 20,  0, 49, 23, 32,  3,
23, 45],
       device='cuda:0')
        TOTAL (correct/total): 830.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.1570, Validation Accuracy: 96.0648 %
Epoch 7, Train Loss: 0.0037, Train Accuracy: 99.8591 %
         ---------------------------------------
        Predicted : tensor([ 3, 94, 73, 96,  0,  3, 54, 65, 44, 84, 75, 16,  1,
5,  0,  0],
       device='cuda:0')
        Result : tensor([ 3, 94, 73, 96,  0,  3, 54, 65, 44, 84, 75, 16,  1,  5,
0,  0],
       device='cuda:0')
        TOTAL (correct/total): 830.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.1641, Validation Accuracy: 96.0648 %

 ---------------------------------------
```










```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 2844), started 3:01:16 ago. (Use '!kill␣
  ↪2844' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

```
La moyenne des accuracy: 382.2685 %
```

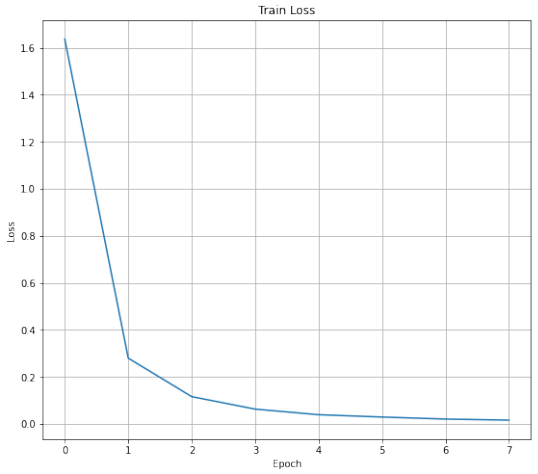
6.5 Pour le modèle densenet161 avec K = 5

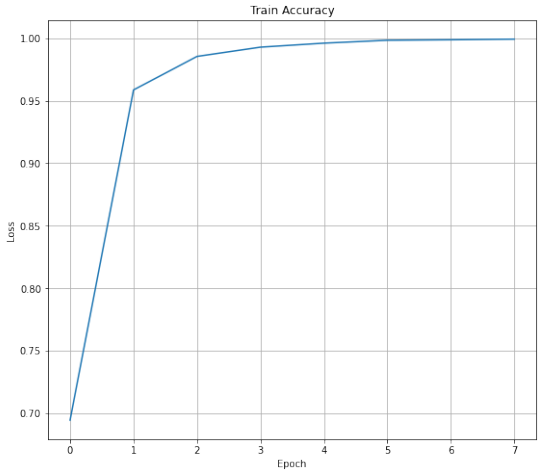
[19]:
```
model_densenet161 = adapt_pretrained_model("densenet161")
loss_fn = nn.CrossEntropyLoss()
optimiseur_densenet161 = torch.optim.SGD(model_densenet161.parameters(), lr=0.
  ↪001, momentum=0.9)
```

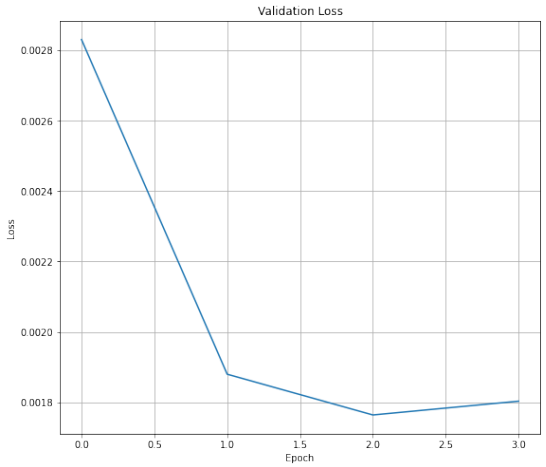
```
mean_densenet161 =␣
 ↪boucle_de_validation_croisee_pour_k_iteration(model_densenet161,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_densenet161, epochs,␣
 ↪typeTrain="densenet161")
```

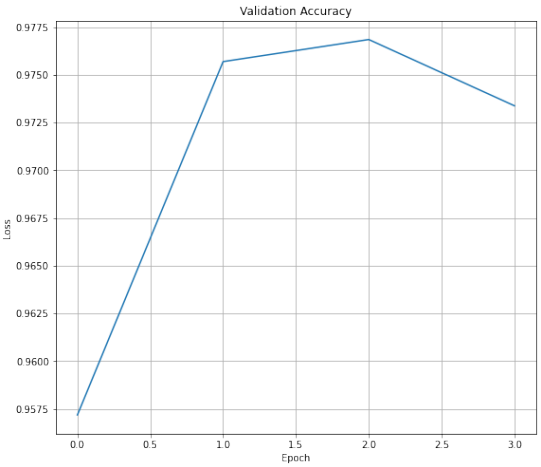
```
Downloading: "https://download.pytorch.org/models/densenet161-8d451a50.pth" to
/root/.cache/torch/hub/checkpoints/densenet161-8d451a50.pth

  0%|          | 0.00/110M [00:00<?, ?B/s]

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 1.6369, Train Accuracy: 69.4672 %
         ----------------------------------------
        Predicted : tensor([ 3, 60,  5,  3, 70,  0, 94, 23,  5, 57, 19, 86,  5,
77,  2, 39],
       device='cuda:0')
        Result : tensor([ 3, 60,  5,  3, 70,  0, 94, 23,  5, 57, 19, 86,  5, 77,
2, 39],
       device='cuda:0')
        TOTAL (correct/total): 827.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.1529, Validation Accuracy: 95.7176 %
Epoch 3, Train Loss: 0.1165, Train Accuracy: 98.5400 %
         ----------------------------------------
        Predicted : tensor([35, 17,  3, 23, 13, 27, 12, 16, 74, 86, 38, 73, 76,
55, 52, 58],
       device='cuda:0')
        Result : tensor([35, 17,  3, 23, 66, 27, 12, 16, 74, 86, 38, 73, 76, 79,
52, 58],
       device='cuda:0')
        TOTAL (correct/total): 843.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.1015, Validation Accuracy: 97.5694 %
Epoch 5, Train Loss: 0.0396, Train Accuracy: 99.6030 %
         ----------------------------------------
        Predicted : tensor([22, 42, 52, 97, 31, 26,  3, 39, 11, 19, 77,  3, 89,
68, 19,  3],
       device='cuda:0')
        Result : tensor([22, 42, 52, 97, 31, 26,  3, 39, 11, 19, 77,  3, 89, 68,
19,  3],
       device='cuda:0')
        TOTAL (correct/total): 844.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.0953, Validation Accuracy: 97.6852 %
```

```
Epoch 7, Train Loss: 0.0211, Train Accuracy: 99.8719 %
         ----------------------------------------
         Predicted : tensor([12,  1,  1, 84, 87, 64, 48,  3, 89, 32,  1, 35, 92,
2,  5,  1],
       device='cuda:0')
         Result : tensor([12,  1,  1, 84, 87, 64, 48,  3, 89, 32,  1, 35, 92,  2,
5,  1],
       device='cuda:0')
         TOTAL (correct/total): 841.0 / 864.0
          ----------------------------------------
         Epoch 8, Validation Loss: 0.0974, Validation Accuracy: 97.3380 %

 ----------------------------------------
```









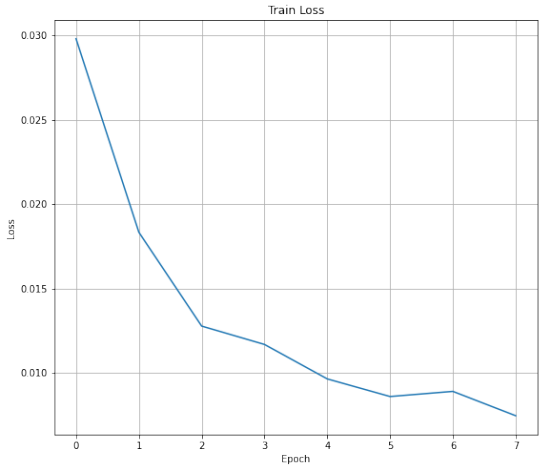

```
The tensorboard extension is already loaded. To reload it, use:
```

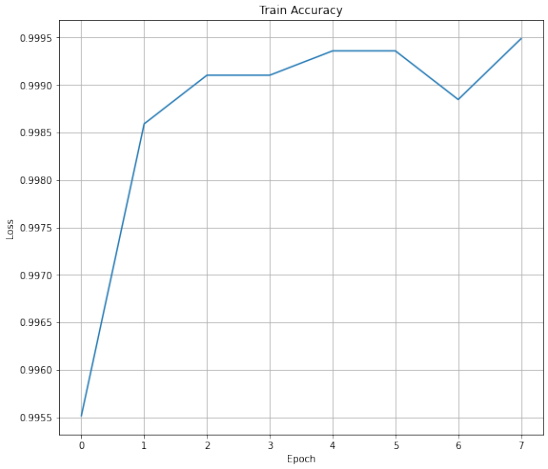
```
  %reload_ext tensorboard
```

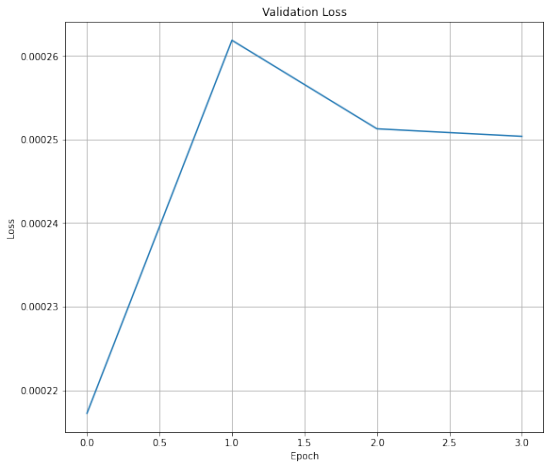
```
<IPython.core.display.Javascript object>
```

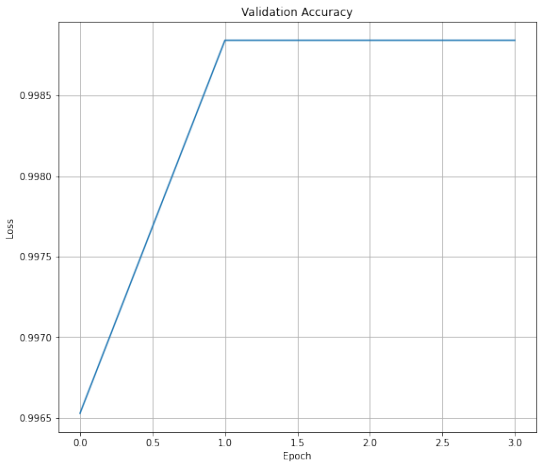
```
La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0298, Train Accuracy: 99.5517 %
         ----------------------------------------
        Predicted : tensor([ 0,  3, 95,  4, 35, 91, 88, 46,  5, 12, 30, 27, 75,
22, 64,  5],
       device='cuda:0')
        Result : tensor([ 0,  3, 95,  4, 35, 91, 88, 46,  5, 12, 30, 27, 75, 22,
64,  5],
       device='cuda:0')
        TOTAL (correct/total): 861.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.0117, Validation Accuracy: 99.6528 %
Epoch 3, Train Loss: 0.0128, Train Accuracy: 99.9103 %
         ----------------------------------------
        Predicted : tensor([34, 94, 50, 38, 18, 93, 44,  1, 53,  6, 67, 93, 17,
25,  1, 66],
       device='cuda:0')
        Result : tensor([34, 94, 50, 38, 18, 93, 44,  1, 53,  6, 67, 93, 17, 25,
1, 66],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.0141, Validation Accuracy: 99.8843 %
Epoch 5, Train Loss: 0.0096, Train Accuracy: 99.9360 %
         ----------------------------------------
        Predicted : tensor([  0,  73, 100, 100,  98,  58,  15,   3,   3,   1,
1,  40,   3,  46,
         35,  24], device='cuda:0')
        Result : tensor([  0,  73, 100, 100,  98,  58,  15,   3,   3,   1,   1,
40,   3,  46,
         35,  24], device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.0136, Validation Accuracy: 99.8843 %
Epoch 7, Train Loss: 0.0089, Train Accuracy: 99.8847 %
         ----------------------------------------
        Predicted : tensor([77,  3,  5, 22,  5,  3, 65,  9,  1,  3, 39, 19, 14,
24, 29, 15],
       device='cuda:0')
        Result : tensor([77,  3,  5, 22,  5,  3, 65,  9,  1,  3, 39, 19, 14, 24,
29, 15],
```

```
      device='cuda:0')
       TOTAL (correct/total): 863.0 / 864.0
        ----------------------------------------
       Epoch 8, Validation Loss: 0.0135, Validation Accuracy: 99.8843 %

----------------------------------------
```





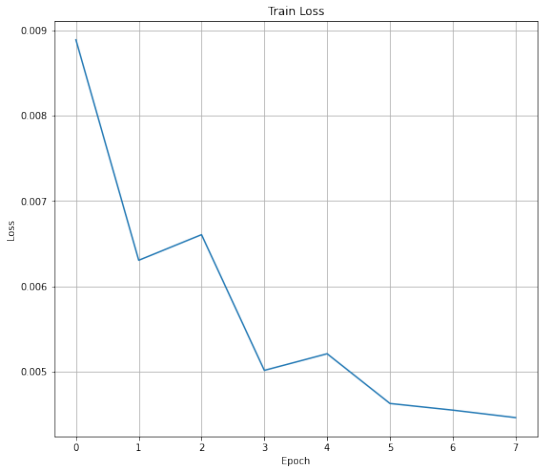


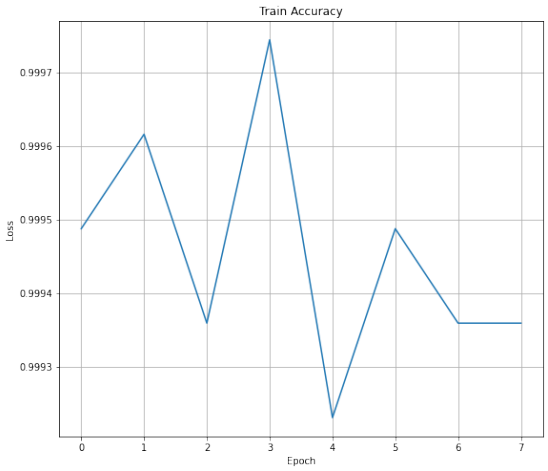


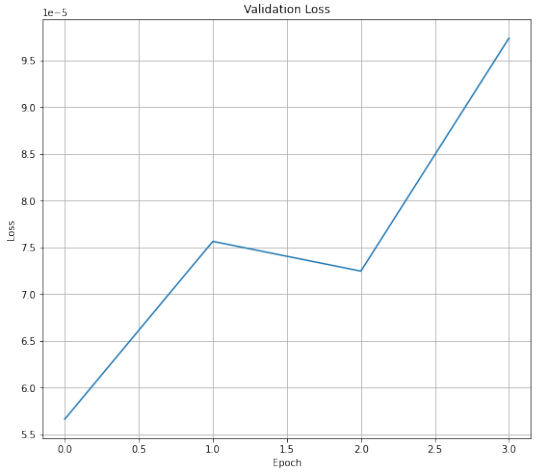

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6007 (pid 34519), started 0:27:34 ago. (Use '!kill␣
  ↪34519' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
```

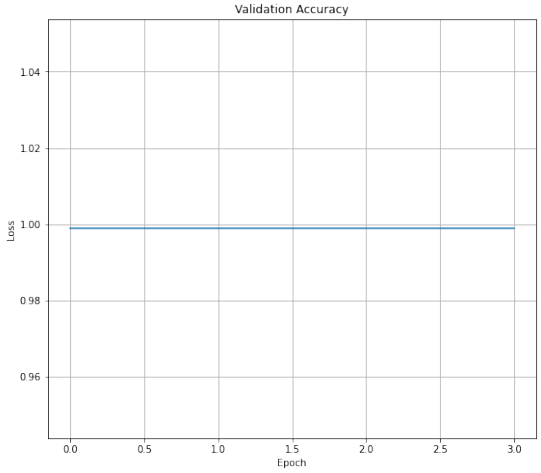
```
Device is : cuda:0

 ---------------------------------------
Epoch 1, Train Loss: 0.0089, Train Accuracy: 99.9488 %
         ---------------------------------------
        Predicted : tensor([ 1, 76, 81, 70,  0, 85, 69, 31, 95, 94, 43,  8,  0,
75, 60, 53],
       device='cuda:0')
        Result : tensor([ 1, 76, 81, 70,  0, 85, 69, 31, 95, 94, 43,  8,  0, 75,
60, 53],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.0031, Validation Accuracy: 99.8843 %
Epoch 3, Train Loss: 0.0066, Train Accuracy: 99.9360 %
         ---------------------------------------
        Predicted : tensor([86,  5, 55, 94, 35, 10,  5, 53, 24, 94,  0, 93, 69,
87,  1, 19],
       device='cuda:0')
        Result : tensor([86,  5, 55, 94, 35, 10,  5, 53, 24, 94,  0, 93, 69, 87,
1, 19],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.0041, Validation Accuracy: 99.8843 %
Epoch 5, Train Loss: 0.0052, Train Accuracy: 99.9232 %
         ---------------------------------------
        Predicted : tensor([35, 21, 47,  5, 86, 92, 89, 34,  3, 87,  4,  1,  7,
3, 94, 96],
       device='cuda:0')
        Result : tensor([35, 21, 47,  5, 86, 92, 89, 34,  3, 87,  4,  1,  7,  3,
94, 96],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.0039, Validation Accuracy: 99.8843 %
Epoch 7, Train Loss: 0.0045, Train Accuracy: 99.9360 %
         ---------------------------------------
        Predicted : tensor([32,  5, 60,  5, 39, 92, 13, 23,  3, 97, 92, 26,  2,
19, 31, 69],
       device='cuda:0')
        Result : tensor([32,  5, 60,  5, 39, 92, 13, 23,  3, 97, 92, 26,  2, 19,
31, 69],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.0053, Validation Accuracy: 99.8843 %
```

```
 ----------------------------------------
```



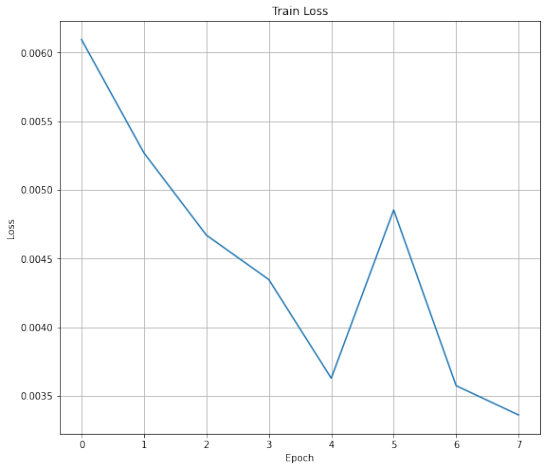


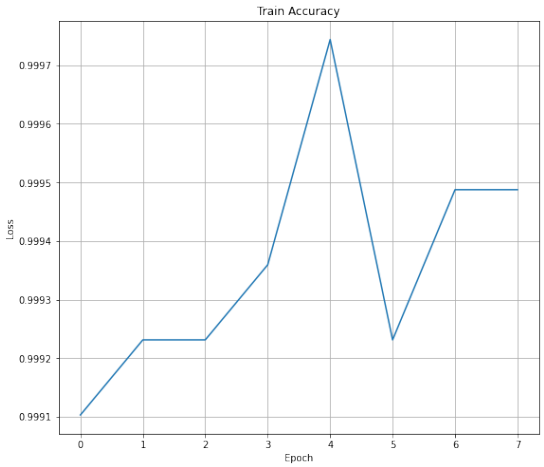


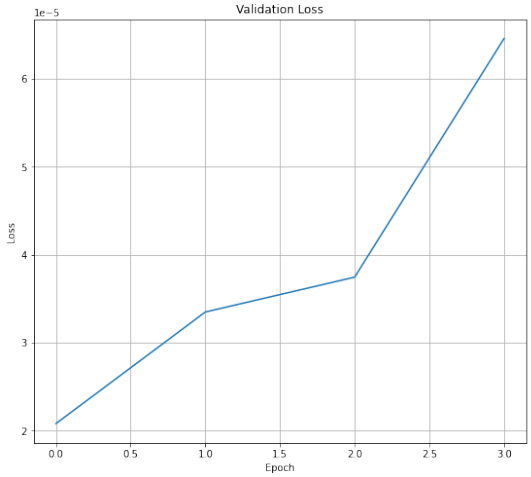


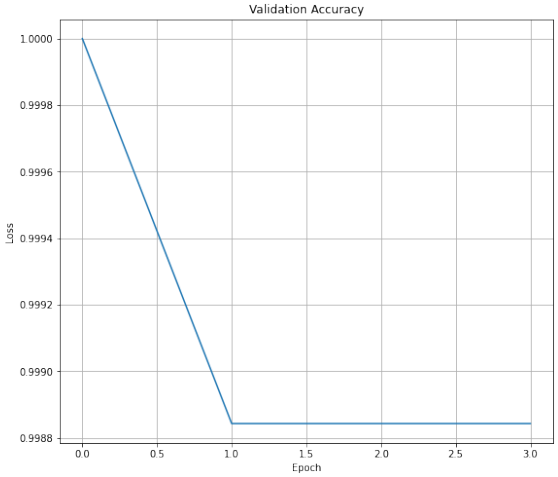

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6007 (pid 34519), started 0:55:26 ago. (Use '!kill␣
 ↪34519' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0061, Train Accuracy: 99.9103 %
         ----------------------------------------
```

```
        Predicted : tensor([19,  3, 38,  1,  0,  0, 27, 86,  5, 23, 23, 36, 92,
62, 90,  5],
       device='cuda:0')
        Result : tensor([19,  3, 38,  1,  0,  0, 27, 86,  5, 23, 23, 36, 92, 62,
90,  5],
       device='cuda:0')
        TOTAL (correct/total): 864.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.0011, Validation Accuracy: 100.0000 %
Epoch 3, Train Loss: 0.0047, Train Accuracy: 99.9232 %
         ----------------------------------------
        Predicted : tensor([56,  3, 58,  3, 38, 96, 54, 92, 83,  5, 13, 40,  1,
31,  2,  0],
       device='cuda:0')
        Result : tensor([56,  3, 58,  3, 38, 96, 54, 92, 83,  5, 13, 40,  1, 31,
2,  0],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ----------------------------------------
        Epoch 4, Validation Loss: 0.0018, Validation Accuracy: 99.8843 %
Epoch 5, Train Loss: 0.0036, Train Accuracy: 99.9744 %
         ----------------------------------------
        Predicted : tensor([27, 63, 48, 78,  0, 30, 19, 58,  3, 93,  1, 20, 38,
5,  5,  1],
       device='cuda:0')
        Result : tensor([27, 63, 48, 78,  0, 30, 19, 58,  3, 93,  1, 20, 38,  5,
5,  1],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ----------------------------------------
        Epoch 6, Validation Loss: 0.0020, Validation Accuracy: 99.8843 %
Epoch 7, Train Loss: 0.0036, Train Accuracy: 99.9488 %
         ----------------------------------------
        Predicted : tensor([31,  0, 58, 34,  3, 15, 92, 33, 23, 79, 70,  5,  1,
28, 43, 35],
       device='cuda:0')
        Result : tensor([31,  0, 58, 34,  3, 15, 92, 33, 23, 79, 70,  5,  1, 28,
43, 35],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.0035, Validation Accuracy: 99.8843 %

 ----------------------------------------
```










```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6007 (pid 34519), started 1:22:58 ago. (Use '!kill␣
↪34519' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

```
La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0
```

```
 ----------------------------------------
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
```

```
<ipython-input-19-4ee7e210f2db> in <module>
      3 optimiseur_densenet161 = torch.optim.SGD(model_densenet161.parameters()␣
 ↪lr=0.001, momentum=0.9)
      4
----> 5 mean_densenet161 =␣
 ↪boucle_de_validation_croisee_pour_k_iteration(model_densenet161,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_densenet161, epochs,␣
 ↪typeTrain="densenet161")

<ipython-input-14-0ca759f2d5cb> in␣
 ↪boucle_de_validation_croisee_pour_k_iteration(model, train_loader,␣
 ↪validation_loader, loss_fn, optimizer, epochs, typeTrain, k)
      6
      7         # Apprentissage des données sur le Train et la Validation
----> 8         _, _, _, _, scores = train(model, train_dataloader,␣
 ↪test_dataloader, loss_fn, optimizer, epochs, typeTrain=typeTrain)
      9         scores_total += scores
     10

<ipython-input-16-4b320ba52a2d> in train(model, train_loader, validation_loader␣
 ↪loss_fn, optimizer, epochs, typeTrain)
     39             _, predicted = torch.max(outputs, 1)
     40             loss.backward()
---> 41             optimizer.step()
     42             train_loss += loss.item()
     43             total += labels.size(0)

/usr/local/lib/python3.8/dist-packages/torch/optim/optimizer.py in␣
 ↪wrapper(*args, **kwargs)
    138                 profile_name = "Optimizer.step#{}.step".format(obj.
 ↪__class__.__name__)
    139                 with torch.autograd.profiler.
 ↪record_function(profile_name):
--> 140                     out = func(*args, **kwargs)
    141                     obj._optimizer_step_code()
    142                     return out

/usr/local/lib/python3.8/dist-packages/torch/optim/optimizer.py in␣
 ↪_use_grad(self, *args, **kwargs)
     21         try:
     22             torch.set_grad_enabled(self.defaults['differentiable'])
---> 23             ret = func(self, *args, **kwargs)
     24         finally:
     25             torch.set_grad_enabled(prev_grad)

/usr/local/lib/python3.8/dist-packages/torch/optim/sgd.py in step(self, closure
    149                         momentum_buffer_list.
 ↪append(state['momentum_buffer'])
```

```
    150
--> 151             sgd(params_with_grad,

    152                 d_p_list,
    153                 momentum_buffer_list,

/usr/local/lib/python3.8/dist-packages/torch/optim/sgd.py in sgd(params,␣
 ↪d_p_list, momentum_buffer_list, has_sparse_grad, foreach, weight_decay,␣
 ↪momentum, lr, dampening, nesterov, maximize)
    200         func = _single_tensor_sgd
    201
--> 202     func(params,

    203          d_p_list,
    204          momentum_buffer_list,

/usr/local/lib/python3.8/dist-packages/torch/optim/sgd.py in␣
 ↪_single_tensor_sgd(params, d_p_list, momentum_buffer_list, weight_decay,␣
 ↪momentum, lr, dampening, nesterov, maximize, has_sparse_grad)
    236                 momentum_buffer_list[i] = buf
    237             else:
--> 238                 buf.mul_(momentum).add_(d_p, alpha=1 - dampening)
    239
    240             if nesterov:

KeyboardInterrupt:
```

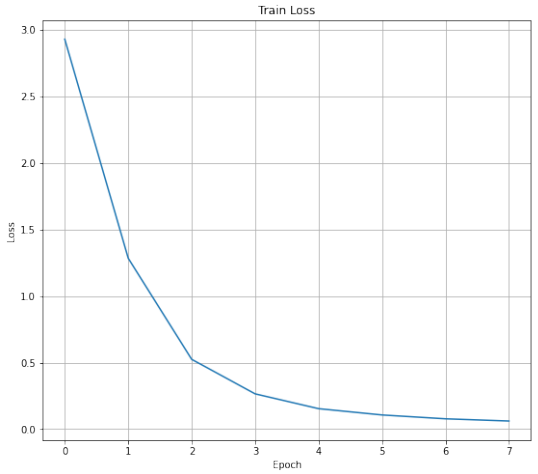
6.6 Pour le modèle inception_v3 avec K = 5

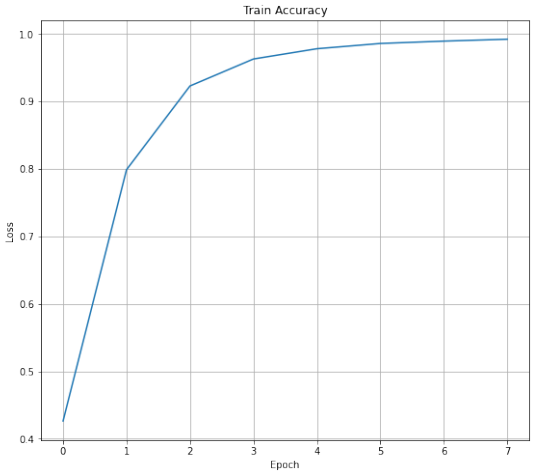
[17]:
```
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
size_of_image = [299, 299]

batch_size = 32
split = 0.9
epochs = 8

transform = transforms.Compose([
    transforms.Resize(size_of_image),
    transforms.ToTensor(),
    transforms.Normalize(mean, std)
])

model_inception_v3 = adapt_pretrained_model("inception_v3")
loss_fn = nn.CrossEntropyLoss()
optimiseur_inception_v3 = torch.optim.SGD(model_inception_v3.parameters(), lr=0.
 ↪001, momentum=0.9)
```

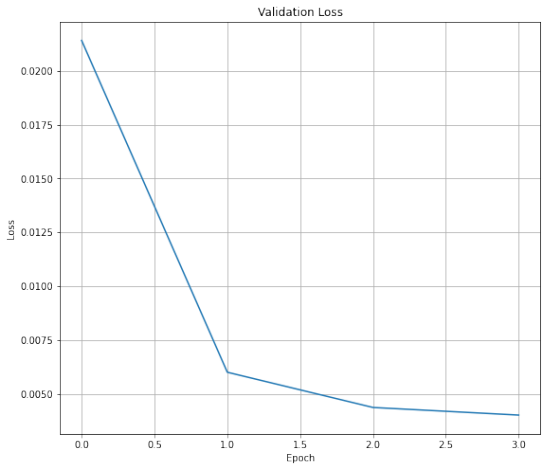
```
mean_inception_v3 =␣
 ↪boucle_de_validation_croisee_pour_k_iteration(model_inception_v3,␣
 ↪train_dataloader, test_dataloader, loss_fn, optimiseur_inception_v3, epochs,␣
 ↪typeTrain="inception_v3")
```

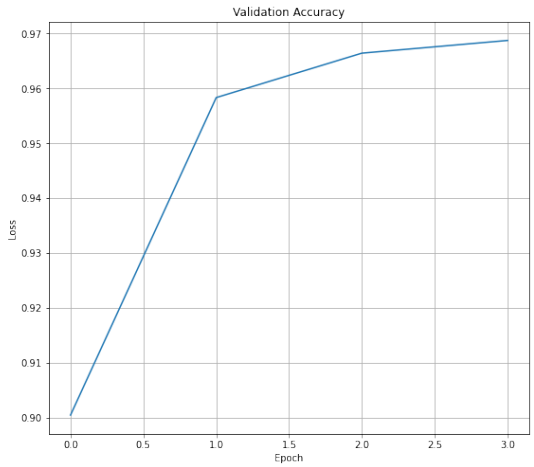
```
La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ---------------------------------------
Epoch 1, Train Loss: 2.9309, Train Accuracy: 42.6230 %
         ---------------------------------------
        Predicted : tensor([ 25,  56,   5,   5,  47,  36,   5,   0,  51,  35,
2,  41,  76,   1,
         37,  35, 100,  16,   5,  13,   5,  72,  16,  96,  17,   1,   3,   5,
         35,   5,   1,  54], device='cuda:0')
        Result : tensor([ 25,  56,   5,   5,  47,  36,   5,   0,  51,  35,   2,
42,  76,   1,
         37,  61, 100,  16,   5,  13,   5,  72,  16,  96,  11,   1,   3,   5,
         35,   5,   1,  14], device='cuda:0')
        TOTAL (correct/total): 778.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.5780, Validation Accuracy: 90.0463 %
Epoch 3, Train Loss: 0.5248, Train Accuracy: 92.2900 %
         ---------------------------------------
        Predicted : tensor([55, 33, 41, 28,  3, 82, 71,  4,  9, 65, 44, 60, 39,
0, 64,  5,  3, 36,
         5,  2,  3, 19,  5,  5, 53, 11,  5, 57, 59, 21, 51,  1],
       device='cuda:0')
        Result : tensor([55, 33, 41, 28,  3, 82, 71,  4, 28, 65, 44, 60, 39,  0,
64,  5,  3, 36,
         5,  2,  3, 19,  5,  5, 53, 11,  5, 57, 59, 21, 51,  1],
       device='cuda:0')
        TOTAL (correct/total): 828.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.1622, Validation Accuracy: 95.8333 %
Epoch 5, Train Loss: 0.1540, Train Accuracy: 97.8099 %
         ---------------------------------------
        Predicted : tensor([ 27,  65,  40,  35,  68,  45,  74,  68,  86,  91,
78,   5,  69,   5,
         47,  19,   5,  28,   0,  47,  97,   1,  47,   3,  23, 100,   1,  91,
         12,   5,   5,  20], device='cuda:0')
        Result : tensor([ 59,  65,  40,  35,  68,  45,  74,  68,  86,  91,  78,
5,  69,   5,
         47,  19,   5,  28,   0,  47,  97,   1,  47,   3,  63, 100,   1,  91,
         12,   5,   5,  20], device='cuda:0')
        TOTAL (correct/total): 835.0 / 864.0
         ---------------------------------------
```

```
        Epoch 6, Validation Loss: 0.1180, Validation Accuracy: 96.6435 %
Epoch 7, Train Loss: 0.0779, Train Accuracy: 98.9370 %
         ----------------------------------------
        Predicted : tensor([28, 68,  3, 76,  5, 45, 20, 16,  3,  0, 58, 76, 94,
3, 47, 56,  1, 47,
        75, 29, 67, 24,  2,  1, 65, 95,  5, 25,  3, 75, 51,  2],
       device='cuda:0')
        Result : tensor([28, 68,  3, 67,  5, 45, 20, 16,  3,  0, 58, 76, 94,  3,
47, 56,  1, 47,
        75, 29,  7, 24,  2,  1, 65, 95,  5, 25,  3, 75, 51,  2],
       device='cuda:0')
        TOTAL (correct/total): 837.0 / 864.0
         ----------------------------------------
        Epoch 8, Validation Loss: 0.1085, Validation Accuracy: 96.8750 %

 ----------------------------------------
```



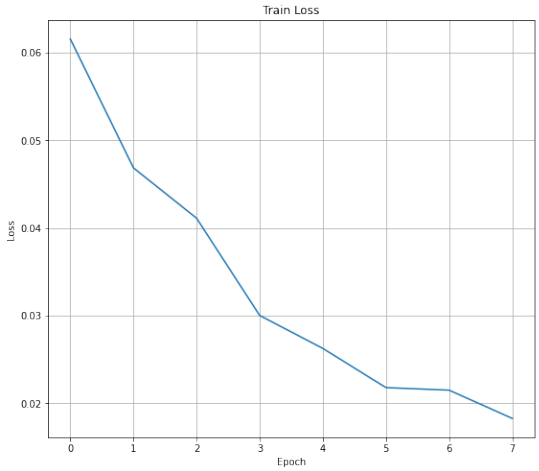


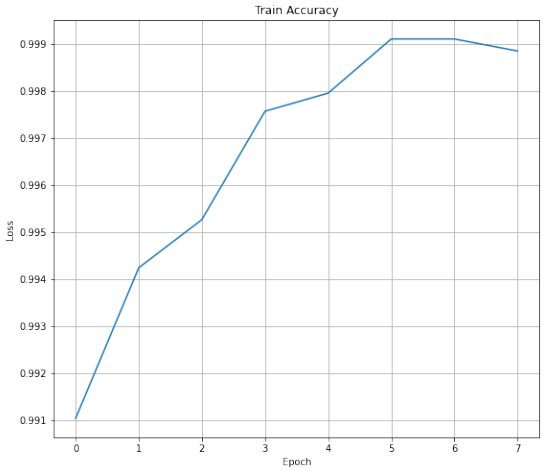


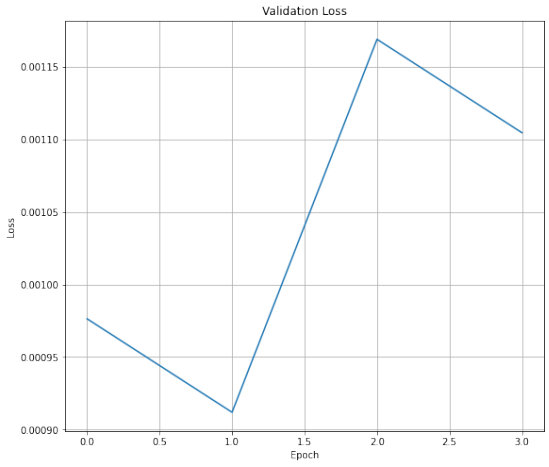


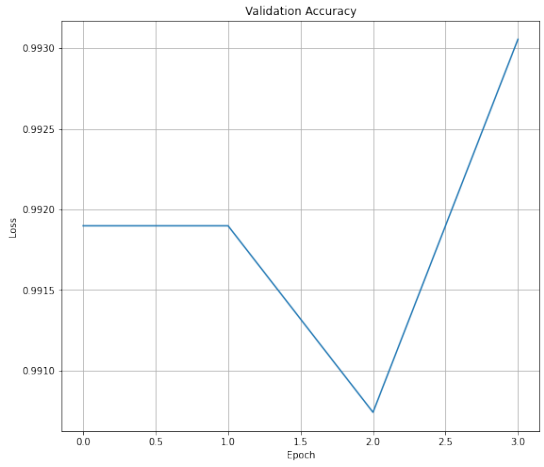
```
<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0616, Train Accuracy: 99.1035 %
        ----------------------------------------
       Predicted : tensor([  3,  51,  87,  16,  29,  46,  55,  12,   2,  47,
94,  94,   2,  54,
         53,  50,   5,  50,  82,  79,  78,  64,   3,  40,  13,   3, 100,   3,
         88,  96,   1,   5], device='cuda:0')
       Result : tensor([  3,  51,  87,  16,  29,  46,  55,  12,   2,  47,  94,
94,   2,  54,
         53,  50,   5,  50,  82,  79,  78,  64,   3,  40,  13,   3, 100,   3,
         88,  96,   1,   5], device='cuda:0')
       TOTAL (correct/total): 857.0 / 864.0
        ----------------------------------------
       Epoch 2, Validation Loss: 0.0264, Validation Accuracy: 99.1898 %
Epoch 3, Train Loss: 0.0411, Train Accuracy: 99.5261 %
        ----------------------------------------
       Predicted : tensor([31,  2, 82, 12,  5,  2, 24,  1, 34, 96,  5, 90, 31,
1,  5, 90,  1, 27,
        29, 87, 55, 65,  7, 94, 63, 74, 28, 26, 92,  8, 23,  1],
      device='cuda:0')
       Result : tensor([31,  2, 82, 12,  5,  2, 24,  1, 34, 96,  5, 90, 31,  1,
5, 90,  1, 27,
        29, 87, 55, 65,  7, 94, 63, 74, 28, 26, 92,  8, 23,  1],
      device='cuda:0')
       TOTAL (correct/total): 857.0 / 864.0
        ----------------------------------------
       Epoch 4, Validation Loss: 0.0246, Validation Accuracy: 99.1898 %
Epoch 5, Train Loss: 0.0263, Train Accuracy: 99.7951 %
        ----------------------------------------
       Predicted : tensor([94, 96, 76, 93, 79,  3, 93, 66, 26, 66,  2, 58, 44,
3,  5, 60,  0,  0,
        73, 21,  1,  5,  0, 25, 13, 21, 68,  3, 30, 64, 93, 15],
      device='cuda:0')
       Result : tensor([94, 96, 76, 93, 79,  3, 93, 66, 26, 66,  2, 58, 44,  3,
5, 60,  0,  0,
        73, 21,  1,  5,  0, 25, 13, 21, 68,  3, 30, 64, 93, 15],
      device='cuda:0')
       TOTAL (correct/total): 856.0 / 864.0
        ----------------------------------------
       Epoch 6, Validation Loss: 0.0316, Validation Accuracy: 99.0741 %
Epoch 7, Train Loss: 0.0215, Train Accuracy: 99.9103 %
```

```
        ----------------------------------------
        Predicted : tensor([13, 58, 18,  5, 69,  6, 25, 55,  5,  1,  3,  3,  3,
97, 12, 57, 99,  3,
         3,  1, 81, 76, 15, 12,  3, 57, 88, 39, 50, 29, 13, 46],
       device='cuda:0')
        Result : tensor([13, 58, 18,  5, 69,  6, 25, 55,  5,  1,  3,  3,  3, 97,
12, 57, 99,  3,
         3,  1, 81, 76, 15, 12,  3, 57, 88, 39, 50, 29, 13, 46],
       device='cuda:0')
        TOTAL (correct/total): 858.0 / 864.0
        ----------------------------------------
        Epoch 8, Validation Loss: 0.0298, Validation Accuracy: 99.3056 %

----------------------------------------
```



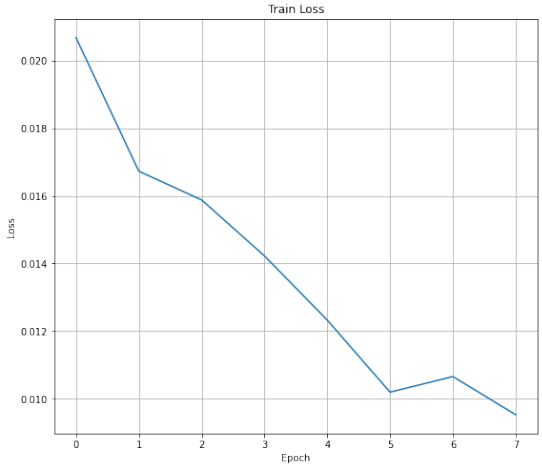


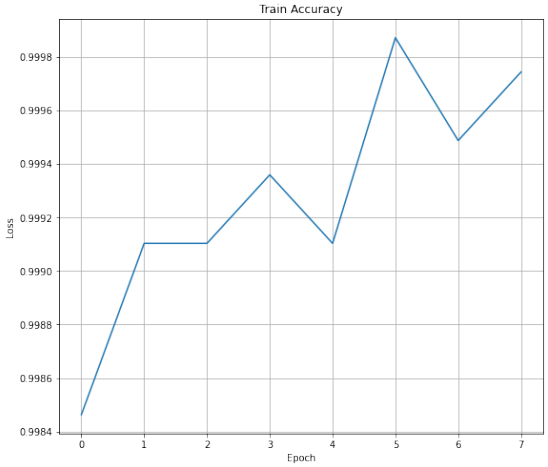


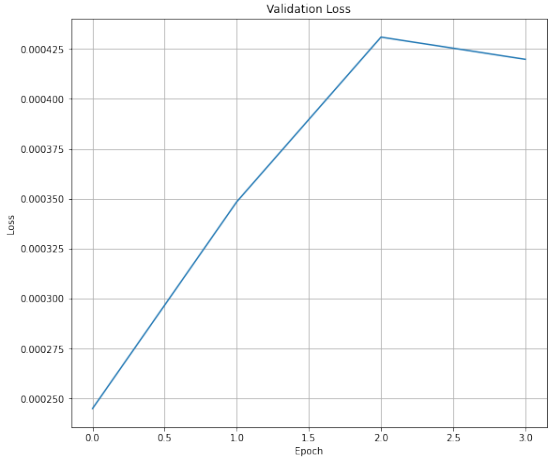


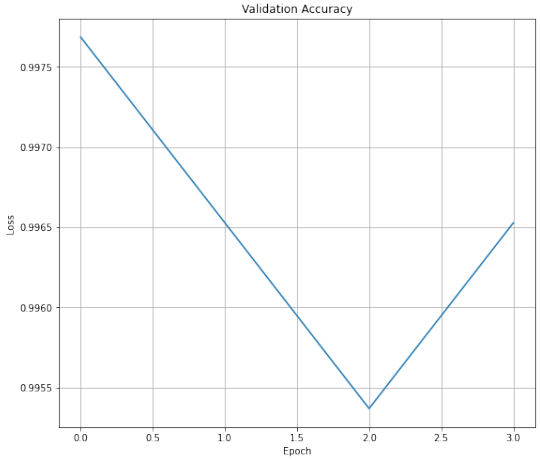
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 7834), started 0:19:19 ago. (Use '!kill␣
 ↪7834' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ---------------------------------------
Epoch 1, Train Loss: 0.0207, Train Accuracy: 99.8463 %
         ---------------------------------------
        Predicted : tensor([ 2,  1,  5, 40, 33,  5,  3, 92,  1, 86, 65,  0,  0,
29,  1,  7,  5, 91,
         5,  0, 29, 29, 21, 94, 26, 86,  0,  1, 73, 22, 58, 12],
       device='cuda:0')
        Result : tensor([ 2,  1,  5, 40, 33,  5,  3, 92,  1, 86, 65,  0,  0, 29,
1,  7,  5, 91,
         5,  0, 29, 29, 21, 94, 26, 86,  0,  1, 73, 22, 58, 12],
       device='cuda:0')
        TOTAL (correct/total): 862.0 / 864.0
         ---------------------------------------
        Epoch 2, Validation Loss: 0.0066, Validation Accuracy: 99.7685 %
Epoch 3, Train Loss: 0.0159, Train Accuracy: 99.9103 %
         ---------------------------------------
        Predicted : tensor([ 41,   0,  31,   5,  27,  51, 100,  27,   3,  95,
66,  16,  23,  46,
          0,  94,   0,  89,   0,   1,  49,  34,   8,  76,  81,   5,   2,   3,
          0,  33,  56,  63], device='cuda:0')
        Result : tensor([ 41,   0,  31,   5,  27,  51, 100,  27,   3,  95,  66,
16,  23,  46,
          0,  94,   0,  89,   0,   1,  49,  34,   8,  76,  81,   5,   2,   3,
          0,  33,  56,  63], device='cuda:0')
        TOTAL (correct/total): 861.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.0094, Validation Accuracy: 99.6528 %
Epoch 5, Train Loss: 0.0123, Train Accuracy: 99.9103 %
         ---------------------------------------
        Predicted : tensor([ 0, 57,  3, 51, 75, 52,  3,  1, 46, 81,  2, 65,  5,
3, 29, 31, 41, 25,
         3, 94,  2, 13, 16,  0,  0, 39,  1,  3, 35,  1, 81, 23],
       device='cuda:0')
        Result : tensor([ 0, 57,  3, 51, 75, 52,  3,  1, 46, 81,  2, 65,  5,  3,
29, 31, 41, 25,
         3, 94,  2, 13, 16,  0,  0, 39,  1,  3, 35,  1, 81, 23],
       device='cuda:0')
```

```
        TOTAL (correct/total): 860.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.0116, Validation Accuracy: 99.5370 %
Epoch 7, Train Loss: 0.0107, Train Accuracy: 99.9488 %
         ---------------------------------------
        Predicted : tensor([ 92,   5,  54,   5,   2,  69,  38,  40,   1,  44,
5,   5,  31,   2,
         37, 100,  83,  64,  27,   3,  12,   3,  53,  75,   0,  14,   0,   0,
         44,  87,  19,  11], device='cuda:0')
        Result : tensor([ 92,   5,  54,   5,   2,  69,  38,  40,   1,  44,   5,
5,  31,   2,
         37, 100,  83,  64,  27,   3,  12,   3,  53,  75,   0,  14,   0,   0,
         44,  87,  19,  11], device='cuda:0')
        TOTAL (correct/total): 861.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.0113, Validation Accuracy: 99.6528 %

 ---------------------------------------
```



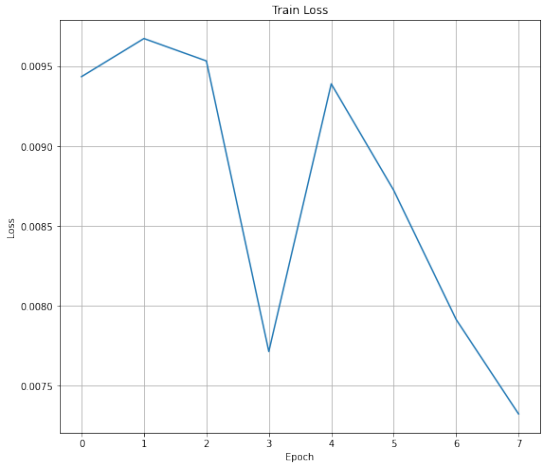


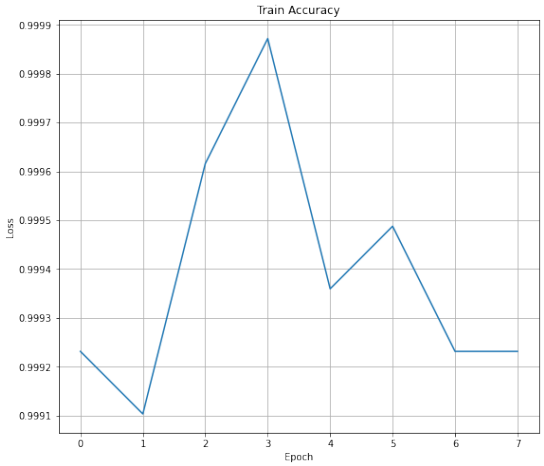


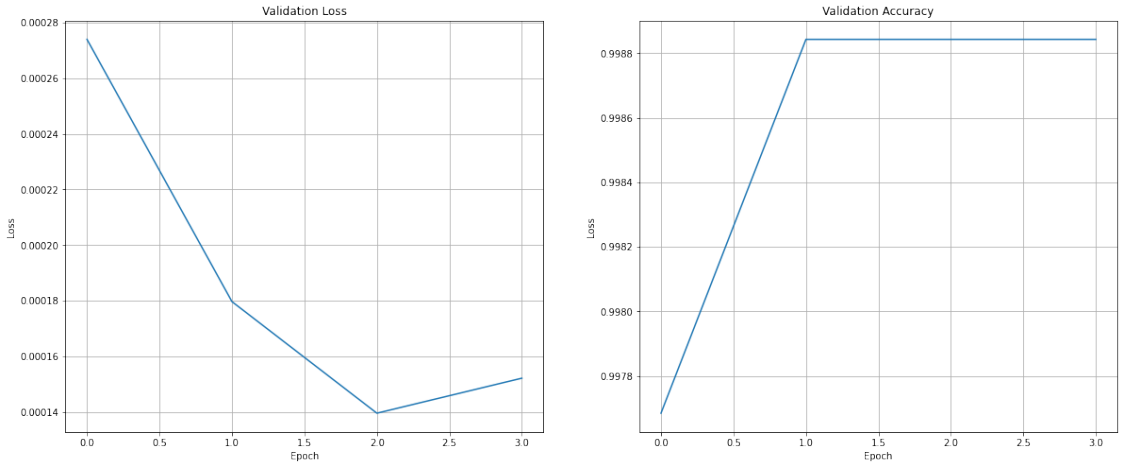


```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 7834), started 0:38:40 ago. (Use '!kill␣
 ↪7834' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0094, Train Accuracy: 99.9232 %
        ----------------------------------------
        Predicted : tensor([ 19,  99,   3,   3,  11,  13,   1,  12,   1,  70,
40,   9,  52,  29,
         46,   5,  41,  22,  91,   5,   5,  30,   2,  49,  98,  25,   3, 100,
         65,   5,  62,  52], device='cuda:0')
        Result : tensor([ 19,  99,   3,   3,  11,  13,   1,  12,   1,  70,  40,
9,  52,  29,
         46,   5,  41,  22,  91,   5,   5,  30,   2,  49,  98,  25,   3, 100,
         65,   5,  62,  52], device='cuda:0')
        TOTAL (correct/total): 862.0 / 864.0
        ----------------------------------------
        Epoch 2, Validation Loss: 0.0074, Validation Accuracy: 99.7685 %
Epoch 3, Train Loss: 0.0095, Train Accuracy: 99.9616 %
        ----------------------------------------
        Predicted : tensor([ 51,  92,   9,   0,   8,   0,  75,  57,   3,  24,
71,   3,   3,   0,
         83,  12,  58,   3,   2, 100,  75,  34,  68,  15,  53,   5,   1,   4,
         44,  67,  58,  56], device='cuda:0')
        Result : tensor([ 51,  92,   9,   0,   8,   0,  75,  57,   3,  24,  71,
3,   3,   0,
         83,  12,  58,   3,   2, 100,  75,  34,  68,  15,  53,   5,   1,   4,
         44,  67,  58,  56], device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
        ----------------------------------------
        Epoch 4, Validation Loss: 0.0049, Validation Accuracy: 99.8843 %
Epoch 5, Train Loss: 0.0094, Train Accuracy: 99.9360 %
        ----------------------------------------
        Predicted : tensor([45, 97, 20,  1,  5, 67, 84, 93, 93,  2, 60, 81, 79,
43,  5, 45, 25,  3,
         3, 94, 13, 94, 86, 38, 94, 49, 79, 12, 46,  5,  5, 31],
       device='cuda:0')
        Result : tensor([45, 97, 20,  1,  5, 67, 84, 93, 93,  2, 60, 81, 79, 43,
```

```
5, 45, 25,  3,
         3, 94, 13, 94, 86, 38, 94, 49, 79, 12, 46,  5,  5, 31],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.0038, Validation Accuracy: 99.8843 %
Epoch 7, Train Loss: 0.0079, Train Accuracy: 99.9232 %
         ---------------------------------------
        Predicted : tensor([ 5,  3,  5, 50,  6,  1, 46, 79,  3, 55, 61, 39, 20,
31,  3, 49, 70,  5,
        50, 27, 97,  3,  1, 59, 33, 14,  3,  3,  1, 25, 66,  3],
       device='cuda:0')
        Result : tensor([ 5,  3,  5, 50,  6,  1, 46, 79,  3, 55, 61, 39, 20, 31,
3, 49, 70,  5,
        50, 27, 97,  3,  1, 59, 33, 14,  3,  3,  1, 25, 66,  3],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.0041, Validation Accuracy: 99.8843 %

 ----------------------------------------
```

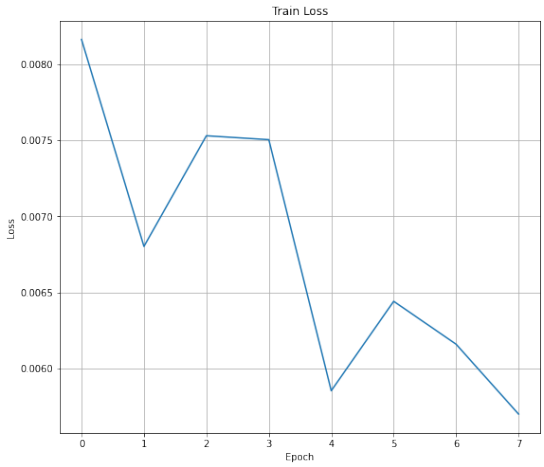


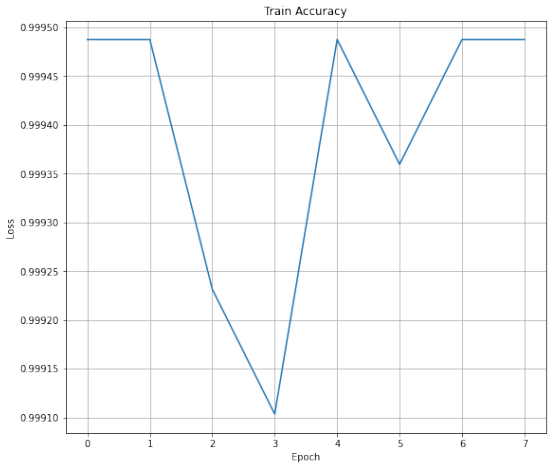


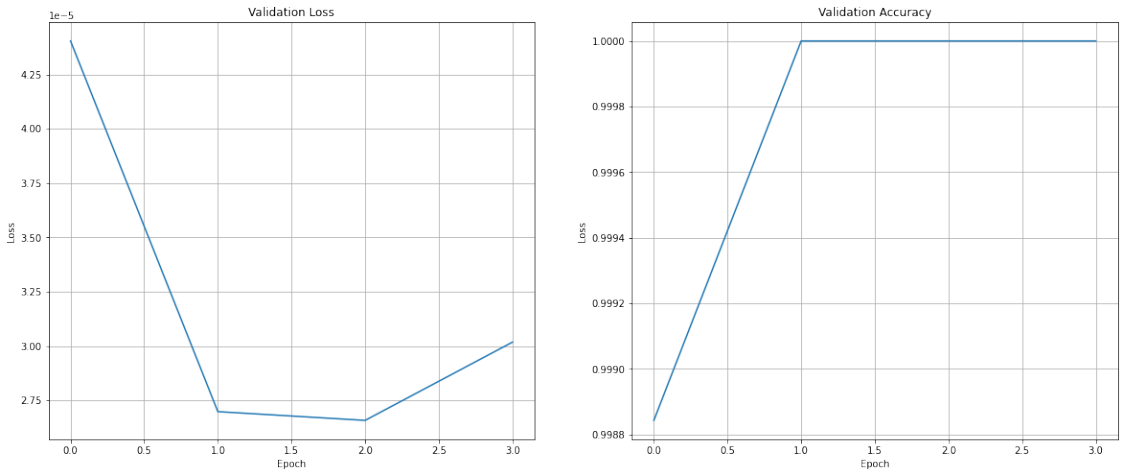



```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 7834), started 0:58:10 ago. (Use '!kill␣
 ↪7834' to kill it.)

<IPython.core.display.Javascript object>

La liste des fichiers du dossier '/content/' sur colab : ['.config',
'caltech-101.zip', '101_ObjectCategories', '__MACOSX', 'sample_data']
Device is : cuda:0

 ----------------------------------------
Epoch 1, Train Loss: 0.0082, Train Accuracy: 99.9488 %
         ----------------------------------------
        Predicted : tensor([35,  5,  5, 19, 23, 51, 27, 96, 40, 13, 54,  7,  5,
28,  2,  1, 34,  3,
        81,  5, 31, 88, 11, 69, 23,  0, 33, 21,  0, 92,  3, 91],
       device='cuda:0')
        Result : tensor([35,  5,  5, 19, 23, 51, 27, 96, 40, 13, 54,  7,  5, 28,
2,  1, 34,  3,
        81,  5, 31, 88, 11, 69, 23,  0, 33, 21,  0, 92,  3, 91],
       device='cuda:0')
        TOTAL (correct/total): 863.0 / 864.0
         ----------------------------------------
        Epoch 2, Validation Loss: 0.0012, Validation Accuracy: 99.8843 %
Epoch 3, Train Loss: 0.0075, Train Accuracy: 99.9232 %
         ----------------------------------------
        Predicted : tensor([ 5, 42,  1, 26, 36, 40,  3,  5,  6,  3, 81,  6, 15,
91,  0,  5,  0, 33,
         3, 50, 22, 59,  5, 19, 13,  0,  1, 87, 92,  3,  5, 11],
       device='cuda:0')
        Result : tensor([ 5, 42,  1, 26, 36, 40,  3,  5,  6,  3, 81,  6, 15, 91,
```

```
0,  5,  0, 33,
         3, 50, 22, 59,  5, 19, 13,  0,  1, 87, 92,  3,  5, 11],
       device='cuda:0')
        TOTAL (correct/total): 864.0 / 864.0
         ---------------------------------------
        Epoch 4, Validation Loss: 0.0007, Validation Accuracy: 100.0000 %
Epoch 5, Train Loss: 0.0059, Train Accuracy: 99.9488 %
         ---------------------------------------
        Predicted : tensor([ 0, 45, 82, 33,  3, 13, 28,  1, 43, 56, 45, 42, 84,
87, 51, 28,  3, 76,
        22, 39, 92,  2,  3, 58,  5, 89,  3, 26,  5, 32, 67, 86],
       device='cuda:0')
        Result : tensor([ 0, 45, 82, 33,  3, 13, 28,  1, 43, 56, 45, 42, 84, 87,
51, 28,  3, 76,
        22, 39, 92,  2,  3, 58,  5, 89,  3, 26,  5, 32, 67, 86],
       device='cuda:0')
        TOTAL (correct/total): 864.0 / 864.0
         ---------------------------------------
        Epoch 6, Validation Loss: 0.0007, Validation Accuracy: 100.0000 %
Epoch 7, Train Loss: 0.0062, Train Accuracy: 99.9488 %
         ---------------------------------------
        Predicted : tensor([ 6,  2,  3, 10,  5,  3,  2,  3, 25, 59, 58, 56,  0,
81, 89, 71,  1, 94,
        97, 33, 31,  5, 50, 55, 46,  5,  3, 78, 60,  5, 55, 74],
       device='cuda:0')
        Result : tensor([ 6,  2,  3, 10,  5,  3,  2,  3, 25, 59, 58, 56,  0, 81,
89, 71,  1, 94,
        97, 33, 31,  5, 50, 55, 46,  5,  3, 78, 60,  5, 55, 74],
       device='cuda:0')
        TOTAL (correct/total): 864.0 / 864.0
         ---------------------------------------
        Epoch 8, Validation Loss: 0.0008, Validation Accuracy: 100.0000 %

 ----------------------------------------
```








```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 7834), started 1:17:57 ago. (Use '!kill␣
↪7834' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

```
La moyenne des accuracy: 394.8148 %
```

6.7 Comparaison des moyennes des modèles

[23]:
```
def compare_means_model(model_name_1, mean_model_1, model_name_2, mean_model_2):
    print("Comparaison entre les modèles : [", model_name_1, ",", mean_model_1␣
↪* 100.0, "% ] VS [", model_name_2, ",", mean_model_2 * 100.0, "% ] ")
    if mean_model_1 > mean_model_2:
        print("\n_____\nLe␣
↪Modèle", model_name_1, "est le plus performant avec une accuracy moyenne de :
↪", mean_model_1 * 100.0, "%!
↪\n_____\n")
    else:
        print("\n_____\nLe␣
↪Modèle", model_name_2, "est le plus performant avec une accuracy moyenne de :
↪", mean_model_2 * 100.0, "%!
↪\n_____\n")
```

Pour : resnet18 VS alexnet

[ ]:
```
compare_means_model("resnet18", mean_resnet18, "alexnet", mean_alexnet)
```

```
Comparaison entre les modèles : [ resnet18 , 388.35648148148147 % ] VS [ alexnet
, 362.29166666666663 % ]


--------------------------------------------------
Le Modèle resnet18 est le plus performant avec une accuracy moyenne de :
388.35648148148147 %!
--------------------------------------------------
```

ResNet18 et AlexNet sont deux architectures de réseau de neurones convolutifs pour la classification d'images. AlexNet est composé de 5 couches de convolution et 3 couches entièrement connectées. Tandis que ResNet18 a une architecture plus profonde que AlexNet avec 18 couches. Il est donc plus complexe et nécessite plus de ressources de calcul.

En termes de performances et en fonction des informations en notre dispositions. On peut conclure que ResNet18 est généralement considéré comme supérieur à AlexNet.

Pour : squeezenet1_0 VS vgg16

[ ]:
```
compare_means_model("squeezenet1_0", mean_squeezenet1_0, "vgg16", mean_vgg16)
```

```
Comparaison entre les modèles : [ squeezenet1_0 , 353.1712962962963 % ] VS [
vgg16 , 382.26851851851853 % ]


--------------------------------------------------
Le Modèle vgg16 est le plus performant avec une accuracy moyenne de :
382.26851851851853 %!
--------------------------------------------------
```

Le modèle VGG16 et le modèle SqueezeNet1.0 sont tous deux des architectures de réseaux de neurones profonds pour la classification d'images. Ils diffèrent dans leur complexité et leur performance. Le modèle VGG16 a une architecture plus profonde et plus complexe que le modèle SqueezeNet1.0. Il a plus de couches et donc plus de paramètres à entraîner, ce qui peut le rendre plus précis mais également plus lent à entraîner.

Donc en résumé, si l'objectif est d'avoir un modèle avec une haute précision et que le temps de formation n'est pas un facteur critique, VGG16 pourrait être une bonne option. Si la rapidité et la légèreté sont des critères importants, SqueezeNet1.0 pourrait être une meilleure option.

Pour : alexnet VS squeezenet1_0

[ ]:
```
compare_means_model("alexnet", mean_alexnet, "squeezenet1_0",␣
↪mean_squeezenet1_0)
```

```
Comparaison entre les modèles : [ alexnet , 362.29166666666663 % ] VS [
squeezenet1_0 , 353.1712962962963 % ]


--------------------------------------------------
Le Modèle alexnet est le plus performant avec une accuracy moyenne de :
362.29166666666663 %!
```

```
--------------------------------------------------
```

Les modèles AlexNet et SqueezeNet1_0 ont des architectures très différentes, ce qui a un impact sur leurs performances et leurs utilisations potentielles. Ils diffèrent sur le nombre de couches, sur la performance ainsi que sur l'utilisation. Cependant, SqueezeNet1_0 est généralement plus rapide que AlexNet, car il utilise des filtres plus petits et des connexions moins denses entre les couches. D'après les données en notre disposition, on peut conclure que Alexnet est bien plus performant que SqueezeNet1_0. Par contre il est dit sur certain site fiable que SqueezeNet1_0 est plus adapté aux appareils mobiles et aux systèmes embarqués. AlexNet, en revanche, peut être plus adapté aux tâches de vision par ordinateur qui nécessitent des représentations plus complexes et des modèles plus profonds.

Donc, en résumé, les modèles AlexNet et SqueezeNet1_0 ont des architectures différentes qui leur permettent de fonctionner de manière optimale pour des cas d'utilisation différents.

Pour : alexnet VS vgg16

[ ]:
```
compare_means_model("alexnet", mean_alexnet, "vgg16", mean_vgg16)
```

```
Comparaison entre les modèles : [ alexnet , 362.29166666666663 % ] VS [ vgg16 ,
382.26851851851853 % ]


--------------------------------------------------
Le Modèle vgg16 est le plus performant avec une accuracy moyenne de :
382.26851851851853 %!
--------------------------------------------------
```

AlexNet et VGG16 sont deux modèles de réseaux de neurones convolutifs populaires pour la classification d'images. La principale différence entre les deux est la profondeur et la complexité du réseau. AlexNet a été l'un des premiers modèles à utiliser une architecture en profondeur avec huit couches de convolution, tandis que VGG16 est plus profond avec seize couches de convolution. Cette différence de profondeur permet à VGG16 de capturer des caractéristiques plus complexes dans les images, ce qui le rend potentiellement plus précis pour la classification. Cependant, cette complexité a un coût en termes de temps et de ressources de calcul. Le modèle VGG16 est plus lent à entraîner et nécessite une quantité significative de mémoire et de puissance de calcul pour être exécuté efficacement.

Pour : resnet18 VS vgg16

[ ]:
```
compare_means_model("resnet18", mean_resnet18, "vgg16", mean_vgg16)
```

```
Comparaison entre les modèles : [ resnet18 , 388.35648148148147 % ] VS [ vgg16 ,
382.26851851851853 % ]


--------------------------------------------------
Le Modèle resnet18 est le plus performant avec une accuracy moyenne de :
388.35648148148147 %!
--------------------------------------------------
```

ResNet-18 et VGG-16 sont deux architectures de réseaux de neurones profonds pour la classification d'images. A travers nos recherches et nos différentes expérimentations sur les données en notre disposition. Nous avons remarqué que ResNet-18 est généralement plus performant que VGG-16 pour la classification d'images, en particulier lorsque les images sont très complexes. Cependant, VGG-16 est souvent préféré pour des tâches de transfert de style, car il permet de mieux conserver le style de l'image source lors de la génération d'une nouvelle image.

Pour : resnet18 VS densenet161

[25]:
```
compare_means_model("resnet18", mean_resnet18, "densenet161", mean_densenet161)
```

```
Comparaison entre les modèles : [ resnet18 , 388.35648148148147 % ] VS [
densenet161 , 399.412738 % ]


------------------------------------------------------
Le Modèle densenet161 est le plus performant avec une accuracy moyenne de :
399.412738 %!
------------------------------------------------------
```

DenseNet161 et ResNet18 sont deux modèles de réseaux de neurones convolutionnels (CNN) largement utilisés pour la classification d'images. DenseNet utilise une architecture de type "densément connectée" où chaque couche est connectée à toutes les couches suivantes dans le réseau. Cette architecture permet d'avoir une propagation de l'information plus fluide à travers le réseau et limite la disparition du gradient, ce qui peut aider à obtenir de meilleurs résultats sur des ensembles de données plus complexes. ResNet, quant à lui, utilise une architecture de type "résiduelle" qui permet de faciliter l'entrainement de réseaux très profonds en résolvant le problème de la disparition du gradient. Les résidus (ou "skip connections") sont utilisés pour permettre à l'information de se propager plus facilement à travers le réseau.

De ce fait, nous pouvons conclure en fonction de nos différents tests et résultats obtenus que DenseNet161 pourrait potentiellement mieux généraliser et obtenir de meilleurs résultats sur des ensembles de données plus complexes, mais nécessiterait également plus de temps et de ressources pour l'entraînement.

6.7.1 Pour : densenet161 VS inception_v3

[26]:
```
compare_means_model("densenet161", mean_densenet161, "inception_v3",␣
↪mean_inception_v3)
```

```
Comparaison entre les modèles : [ densenet161 , 399.412738 % ] VS [ inception_v3
, 394.81481481481484 % ]


------------------------------------------------------
Le Modèle densenet161 est le plus performant avec une accuracy moyenne de :
399.412738 %!
------------------------------------------------------
```

Les modèles DenseNet161 et Inception_v3 sont tous deux des réseaux de neurones convolutifs profonds pré-entraînés destinés à la classification d'images. DenseNet161 utilise une architecture dense

où chaque couche reçoit en entrée toutes les cartes de caractéristiques des couches précédentes, tandis qu'Inception_v3 utilise une architecture en parallèle qui combine les cartes de caractéristiques de différentes tailles. DenseNet161 utilise des connexions résiduelles pour permettre un apprentissage plus profond et plus facile, tandis qu'Inception_v3 utilise des blocs inception pour extraire des informations à différentes échelles. DenseNet161 a généralement une meilleure précision de classification que Inception_v3 sur des ensembles de données plus petits, mais Inception_v3 est généralement plus rapide et plus léger.

En fin de compte, le choix entre DenseNet161 et Inception_v3 dépend de l'application spécifique et des exigences en matière de précision, de vitesse et de ressources informatiques.

7 Conlusion

Les modèles que nous avons eu à étudier sont des architectures de réseaux de neurones convolutifs populaires dans l'apprentissage en profondeur. Ils diffèrent sur plusieurs plans tels que : la profondeur, le nombre de paramètres, la taille (nombre de couches), l'architecture ainsi que sur la performances des modèles.

Malgré les nombreuses variations au niveau des courbes de nos modèles lors de l'usage de la boucle d'apprentissage. On peut observer que ces variations sont tout a fait normale car ils tournent autour des valeurs extremements bonnes. Nous pouvons conclure que parmi les six (6) modèles (resnet18, alexnet, squeezenet1_0, vgg16, densenet161 et inception_v3). Les modèles les plus performant en fonction des donnnées que nous pouvons exploiter sur nos sorties consoles et graphes, par ordre croissant sont : * densenet161 (1er) * inception_v3 (2e) * resnet18 (3e) * vgg16 (4e) * alexnet (5e) * squeezenet1_0 (6e)

N.B. A la suite d'un manque de ressources du GPU sur google colab, nous avons donc décidé d'utiliser une boucle d'apprentissage pour k = 5 afin de pouvoir tester tous nos modèles. Pour ce faire, il nous a fallut utiliser plusieurs compte gmail afin d'acquérir assez de ressources sur la plateforme. Ceci a été notre seul frein malheureusement. Mais vous avez la possibilité de modifier le paramètre k de notre fonction à tout moment si cela est votre souhait.