IFT780 – TP3

Question 2

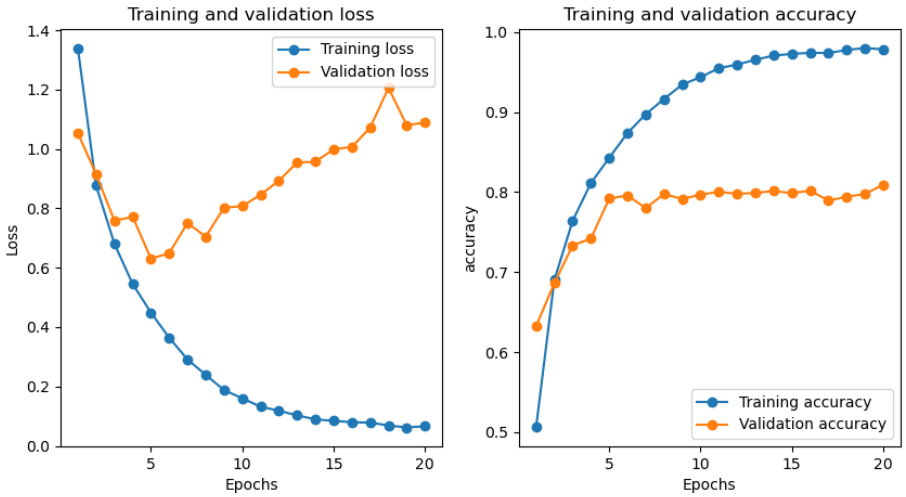
Endroits où du code a été ajouté (texte en vert: ajout, texte en rouge: retrait)

- Fichier ‘src/train.py’:

```
[…]

 from copy import copy

[…]

     if data_augment:
         print('Data augmentation activated!')
         data_augment_transforms = [
             transforms.RandomRotation(15),
             transforms.ColorJitter(contrast=0.1,
                                    hue=0.1),
             transforms.RandomHorizontalFlip(p=0.5),
             transforms.RandomCrop(32, padding=4)
         ]
     else:
         print('Data augmentation NOT activated!')
         data_augment_transforms = []

[…]

    base_transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])

    train_transform = transforms.Compose([
        *data_augment_transforms,
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])

    if args.dataset == 'cifar10':
        # Download the train and test set and apply transform on it
        train_set = datasets.CIFAR10(root='../data', train=True, download=True, transform=base_transform)
        train_set = datasets.CIFAR10(root='../data', train=True, download=True, transform=train_transform)
        test_set = datasets.CIFAR10(root='../data', train=False, download=True, transform=base_transform)

    elif args.dataset == 'svhn':
        # Download the train and test set and apply transform on it
        train_set = datasets.SVHN(root='../data', split='train', download=True, transform=base_transform)
        train_set = datasets.SVHN(root='../data', split='train', download=True, transform=train_transform)
        test_set = datasets.SVHN(root='../data', split='test', download=True, transform=base_transform)

    if val_set:
        len_val_set = int(len(train_set) * val_set)
        train_set, val_set = torch.utils.data.random_split(train_set, [len(train_set) - len_val_set, len_val_set])
        val_set.dataset = copy(train_set.dataset)
        val_set.dataset.transform = base_transform

[…]
```

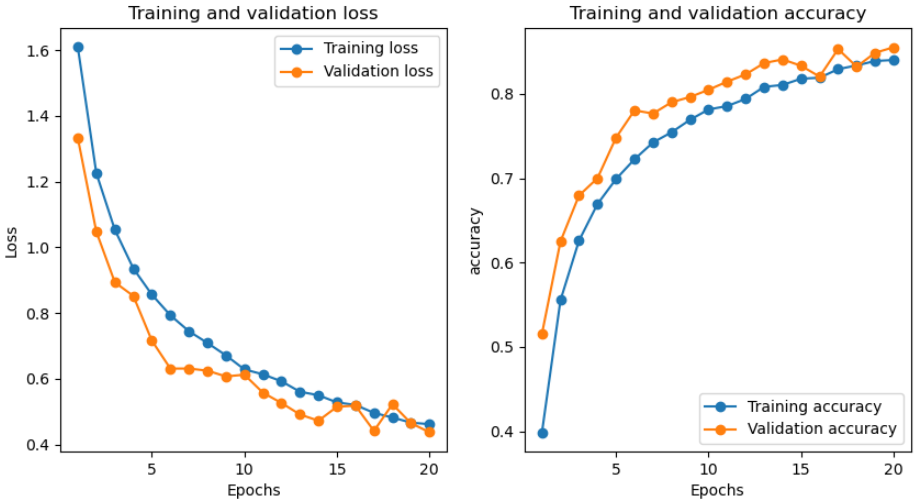
Courbes d’entraînement et de validation

1. --model=CnnVanilla --dataset=cifar10 --num-epochs=20 --batch_size=100




```
(ift725) simon@alien:~/tp3/src$ python train.py --model=CnnVanilla --dataset=cifar10 --num-epochs=20 --batch_size=100
Data augmentation NOT activated!
Files already downloaded and verified
Files already downloaded and verified
Training CnnVanilla on cifar10 for 20 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 80.760 %
```

2. --model=CnnVanilla --dataset=cifar10 --num-epochs=20 --batch_size=100 --data_aug

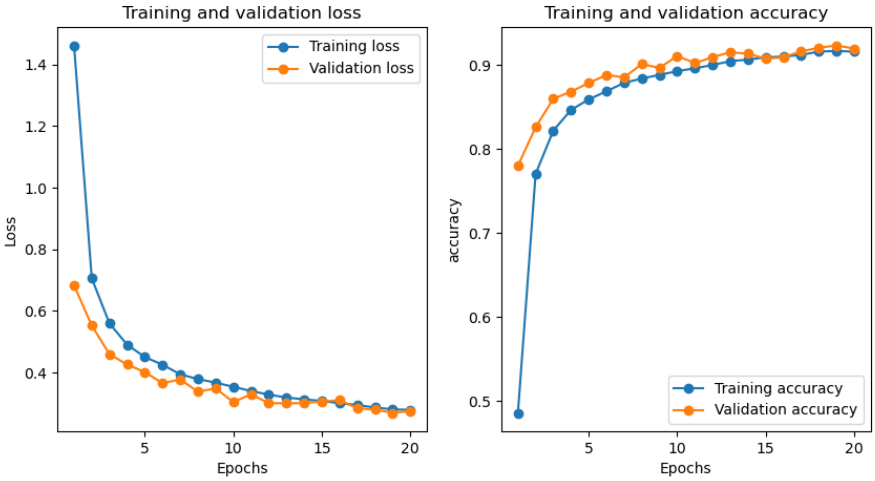



```
(ift725) simon@alien:~/tp3/src$ python train.py --model=CnnVanilla --dataset=cifar10 --num-epochs=20 --batch_size=100 --
data_aug
Data augmentation activated!
Files already downloaded and verified
Files already downloaded and verified
Training CnnVanilla on cifar10 for 20 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 82.780 %
```

3. --model=CnnVanilla --dataset=svhn --num-epochs=20 --batch_size=100


```
((ift725) simon@alien:~/tp3/src$ python train.py --model=CnnVanilla --dataset=svhn --num-epochs=20 --batch_size=100
Data augmentation NOT activated!
Using downloaded and verified file: ../data/train_32x32.mat
Using downloaded and verified file: ../data/test_32x32.mat
Training CnnVanilla on svhn for 20 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 90.627 %
```

4. --model=CnnVanilla --dataset=svhn --num-epochs=20 --batch_size=100 --data_aug




```
(ift725) simon@alien:~/tp3/src$ python train.py --model=CnnVanilla --dataset=svhn --num-epochs=20 --batch_size=100 --
data_aug
Data augmentation activated!
Using downloaded and verified file: ../data/train_32x32.mat
Using downloaded and verified file: ../data/test_32x32.mat
Training CnnVanilla on svhn for 20 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 82.636 %
```

Question 3

Représentation graphique du réseau IFT725Net

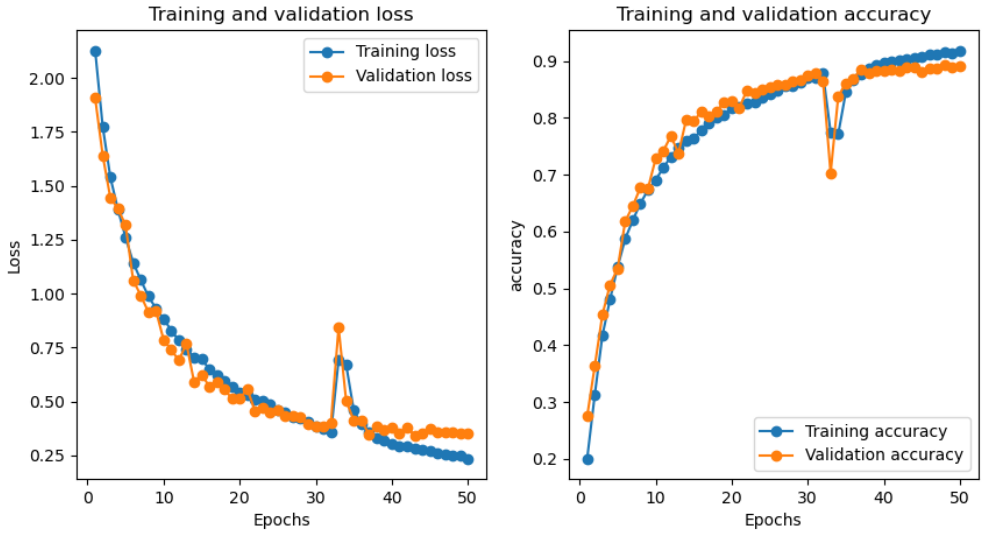
Une représentation graphique du réseau a été créée à l'aide du package *hiddenlayer*. Le code suivant a été ajouté dans le fichier *'src/train.py'*:

```
[…]

import hiddenlayer as hl
import os

[…]

   transforms = [
        hl.transforms.Fold("Conv > BatchNorm > Relu", "ConvBatchNormRelu"),
        hl.transforms.Fold("BatchNorm > Relu > Conv", "BnReluConv"),
        hl.transforms.Prune("Constant"),
        hl.transforms.Prune("Transpose"),
        hl.transforms.Prune("Transpose"),
        hl.transforms.Fold("Reshape > MatMul", "FullyConnected", "FullyConnectedBlock"),
        hl.transforms.Fold("MatMul", "FullyConnected", "FullyConnectedBlock"),
        hl.transforms.Fold("BatchNorm > Relu", "BnRelu"),
    ]

    g = hl.build_graph(model, torch.zeros([1, 3, 32, 32]), transforms=transforms)
    g.save(f'{os.getcwd()}/model', format='pdf')

[…]
```

Voir la figure à l'Annexe 1.

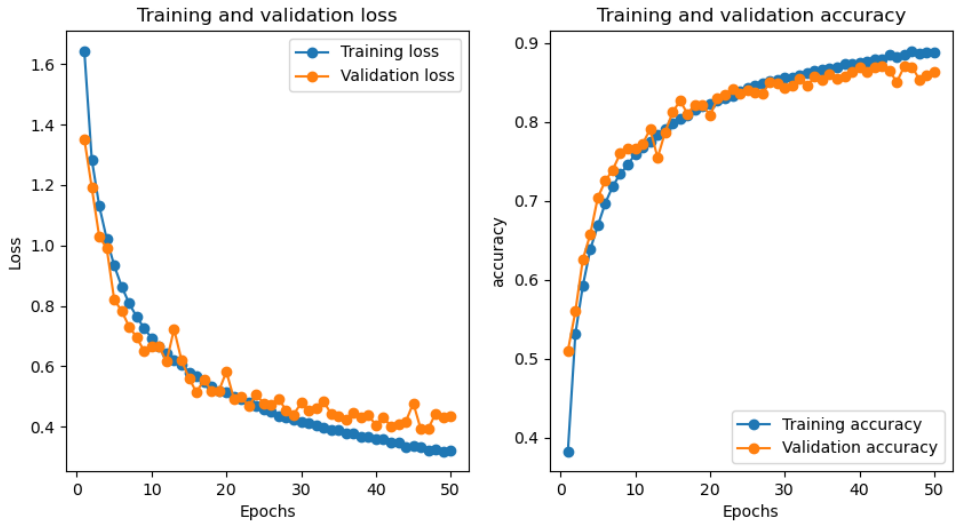
Courbes d’apprentissage et de validation

1. --model=IFT725Net --dataset=cifar10 --num-epochs=50 --batch_size=100 --data_aug




```
(ift725) simon@alien:~/tp3/src$ python train.py --model=IFT725Net --dataset=cifar10 --num-epochs=50 --batch_size=100 --
data_aug
Data augmentation activated!
Files already downloaded and verified
Files already downloaded and verified
Training IFT725Net on cifar10 for 50 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 85.680 %
```

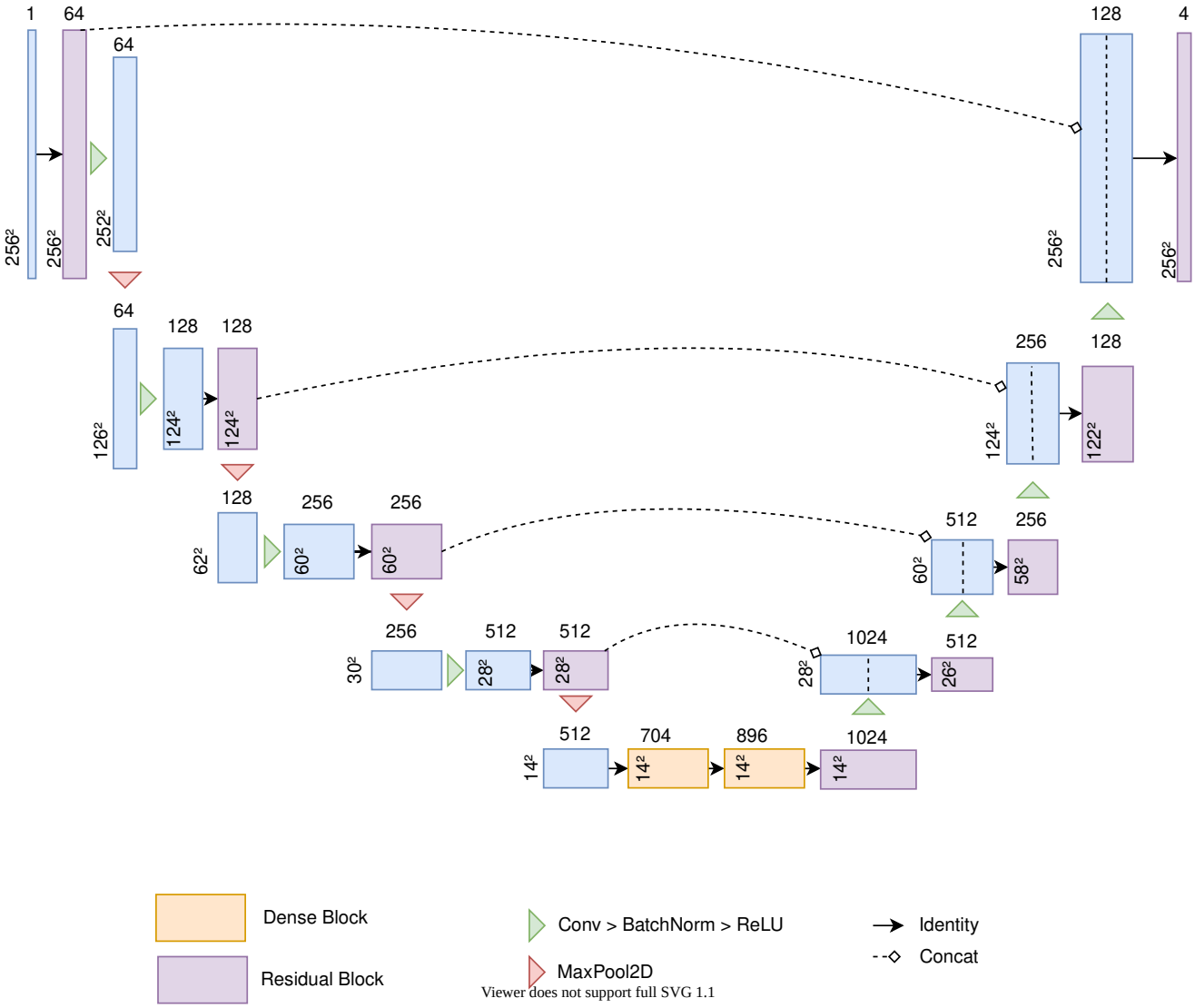
2. -model=CnnVanilla --dataset=cifar10 --num-epochs=50 --batch_size=100 --data_aug




```
(ift725) simon@alien:~/tp3/src$ python train.py --model=CnnVanilla --dataset=cifar10 --num-epochs=50 --batch_size=100 --
data_aug
Data augmentation activated!
Files already downloaded and verified
Files already downloaded and verified
Training CnnVanilla on cifar10 for 50 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 84.540 %
```

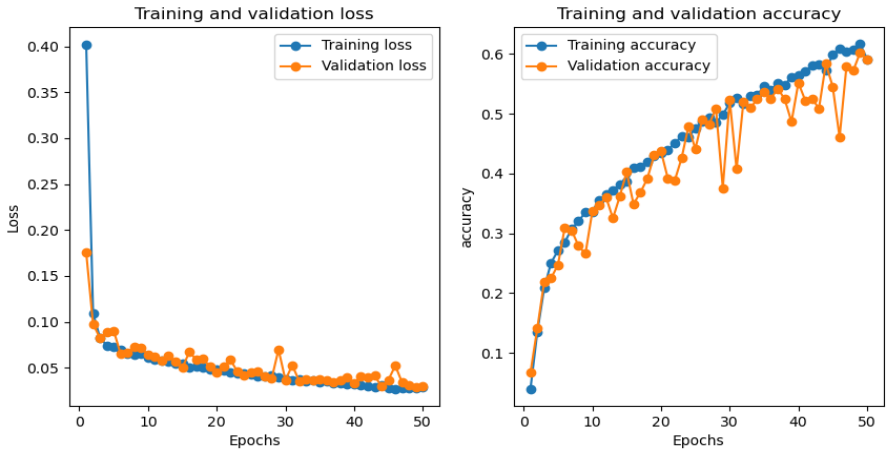
Question 4

Architecture du réseau IFT725UNet



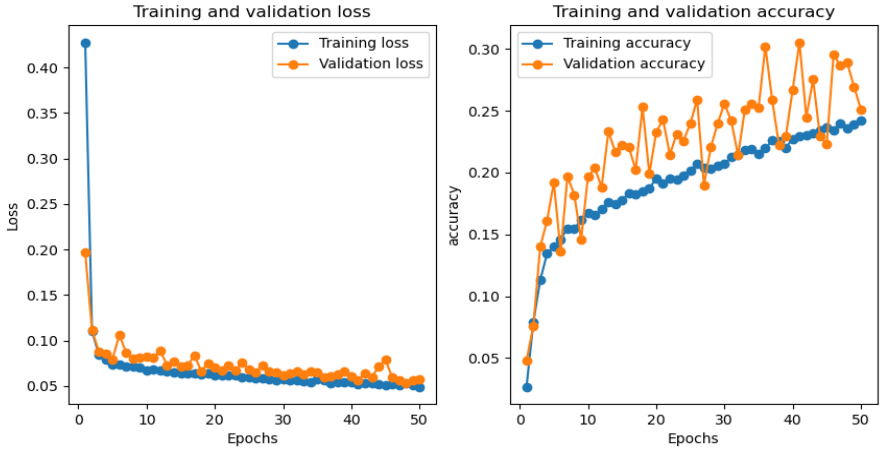
Courbes d’apprentissage et de validation

1. --model=IFT725UNet --num-epochs=50 –batch_size=25




```
python train.py --model=IFT725UNet --num-epochs=50 -batch_size=25
Data augmentation NOT activated!
Files already downloaded and verified
Files already downloaded and verified
Training IFT725UNet on acdc for 50 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 50.878 %
```

2. --model=IFT725UNet --num-epochs=50 --batch_size=25 –data_aug




```
python train.py --model=IFT725UNet --num-epochs=50 --batch_size=25 -data_aug
Data augmentation activated!
Files already downloaded and verified
Files already downloaded and verified
Training IFT725UNet on acdc for 50 epochs

Finished training.
Accuracy (or Dice for UNet) on the test set: 23.718 %
```

Endroits où du code a été ajouté (texte en vert: ajout, texte en rouge: retrait)

- fichier src/CNNTrainTestManager.py

```
[…]

+from models.IFT725UNet import IFT725UNet
 from utils import mean_dice, convert_mask_to_rgb_image
 import matplotlib.pyplot as plt

[…]

@@ -171,7 +172,7 @@ class CNNTrainTestManager(object):
         Returns:
             Accuracy of the model
         """
-        if isinstance(self.model, UNet):
+        if isinstance(self.model, UNet) or isinstance(self.model, IFT725UNet):
             # compute the mean of the 3 classes's dice score
             return mean_dice(outputs, labels).item()
         else:

[…]
```

- fichier src/HDF5Dataset.py

```
[…]

@@ -86,6 +86,8 @@ class HDF5Dataset(data.Dataset):
         summed = np.clip(gt_slice[:, :, 1:].sum(axis=-1), 0, 1)
         gt_slice[:, :, 0] = np.abs(1 - summed)

+        gt_slice = np.argmax(gt_slice, axis=2).astype(dtype='float32')
+
         if self.transform is not None:
             # transform numpy array using the provided transform
             # To apply the same transformations (RandomFlip, RandomCrop) on both image and ground truth

[…]

@@ -105,7 +107,7 @@ class HDF5Dataset(data.Dataset):
             gt_slice = torch.from_numpy(gt_slice)
         # ground truth must be encoded in categorical form instead of one_hot vector
         # in order to allow pytorch's CrossEntropy to compute the loss
-        return img_slice, np.argmax(gt_slice, axis=0)
+        return img_slice, torch.squeeze(gt_slice).type(torch.long)

[…]
```

- fichier src/train.py

```
[…]

 from models.VggNet import VggNet
 from torchvision import datasets

 from copy import copy
+import numpy as np
+from random import uniform


 def argument_parser():
@@ -40,7 +43,7 @@ def argument_parser():
                                                 " need to provide a dataset since UNet model only train "
                                                 "on acdc dataset.")
     parser.add_argument('--model', type=str, default="CnnVanilla",
-                        choices=["CnnVanilla", "VggNet", "AlexNet", "ResNet", "IFT725Net", "UNet"])
+                        choices=["CnnVanilla", "VggNet", "AlexNet", "ResNet", "IFT725Net", "UNet", "IFT725UNet"])
     parser.add_argument('--dataset', type=str, default="cifar10", choices=["cifar10", "svhn"])
     parser.add_argument('--batch_size', type=int, default=20,
                         help='The size of the training batch')

[…]
```

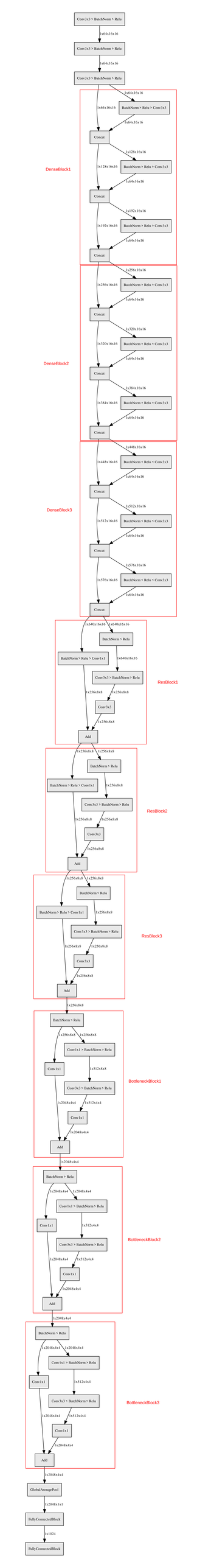
```
@@ -70,13 +73,23 @@ if __name__ == "__main__":
     data_augment = args.data_aug
     if data_augment:
         print('Data augmentation activated!')
-        data_augment_transforms = [
-            transforms.RandomRotation(15),
-            transforms.ColorJitter(contrast=0.1,
-                                   hue=0.1),
-            transforms.RandomHorizontalFlip(p=0.5),
-            transforms.RandomCrop(32, padding=4)
-        ]
+        if args.model == 'UNet' or args.model == "IFT725UNet":
+            data_augment_transforms = [
+                transforms.Lambda(lambda img: img * uniform(0.9, 1.1)),
+                transforms.ToPILImage(),
+                transforms.RandomRotation(15),
+                transforms.RandomCrop(256),
+                transforms.Lambda(lambda img: img.convert('F')),
+                np.array
+            ]
+        else:
+            data_augment_transforms = [
+                transforms.RandomRotation(15),
+                transforms.ColorJitter(contrast=0.1,
+                                       hue=0.1),
+                transforms.RandomHorizontalFlip(p=0.5),
+                transforms.RandomCrop(32, padding=4)
+            ]
+

[…]


+    acdc_train_transform = transforms.Compose([
+        *data_augment_transforms,
+        transforms.ToTensor()
+    ])
+
     base_transform = transforms.Compose([
         transforms.ToTensor(),
         transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))

[…]

@@ -110,12 +128,6 @@ if __name__ == "__main__":
         train_set = datasets.SVHN(root='../data', split='train', download=True, transform=train_transform)
         test_set = datasets.SVHN(root='../data', split='test', download=True, transform=base_transform)

-    if val_set:
-        len_val_set = int(len(train_set) * val_set)
-        train_set, val_set = torch.utils.data.random_split(train_set, [len(train_set) - len_val_set, len_val_set])
-        val_set.dataset = copy(train_set.dataset)
-        val_set.dataset.transform = base_transform
-

[…]

         model = UNet(num_classes=4)
         args.dataset = 'acdc'

-        train_set = HDF5Dataset('train', hdf5_file, transform=acdc_base_transform)
+        train_set = HDF5Dataset('train', hdf5_file, transform=acdc_train_transform)
         test_set = HDF5Dataset('test', hdf5_file, transform=acdc_base_transform)
+    elif args.model == 'IFT725UNet':
+        model = IFT725UNet(num_classes=4)
+        args.dataset = 'acdc'
+
+        train_set = HDF5Dataset('train', hdf5_file, transform=acdc_train_transform)
+        test_set = HDF5Dataset('test', hdf5_file, transform=acdc_base_transform)
+
+    if val_set:
+        len_val_set = int(len(train_set) * val_set)
+        train_set, val_set = torch.utils.data.random_split(train_set, [len(train_set) - len_val_set, len_val_set])
+        val_set.dataset = copy(train_set.dataset)
+        val_set.dataset.transform = acdc_base_transform if args.model == 'UNet' or args.model == 'IFT725UNet' \
+            else base_transform

[…]
```

```
@@ -156,7 +181,7 @@ if __name__ == "__main__":
         print("Training {} on {} for {} epochs".format(model.__class__.__name__, args.dataset, args.num_epochs))
         model_trainer.train(num_epochs)
         model_trainer.evaluate_on_test_set()
-        if isinstance(model, UNet):
+        if isinstance(model, UNet) or isinstance(model, IFT725UNet):
             model.save()  # save the model's weights for prediction (see help for more details)
             model_trainer.plot_image_mask_prediction()
         model_trainer.plot_metrics()
```

ANNEXE 1

Question 3

Représentation graphique du réseau IFT725Net


Conv3x3 > BatchNorm > Relu
1x64x16x16
Conv3x3 > BatchNorm > Relu
1x64x16x16
Conv3x3 > BatchNorm > Relu
DenseBlock1
1x64x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
1x128x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
1x192x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
DenseBlock2
1x256x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
1x320x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
1x384x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
DenseBlock3
1x448x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
1x512x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
1x576x16x16
BatchNorm > Relu > Conv3x3
1x64x16x16
Concat
ResBlock1
1x640x16x16
1x640x16x16
BatchNorm > Relu
BatchNorm > Relu > Conv1x1
1x640x16x16
Conv3x3 > BatchNorm > Relu
1x256x8x8
1x256x8x8
Conv3x3
1x256x8x8
Add
ResBlock2
1x256x8x8
1x256x8x8
BatchNorm > Relu
BatchNorm > Relu > Conv1x1
1x256x8x8
Conv3x3 > BatchNorm > Relu
1x256x8x8
1x256x8x8
Conv3x3
1x256x8x8
Add
ResBlock3
1x256x8x8
1x256x8x8
BatchNorm > Relu
BatchNorm > Relu > Conv1x1
1x256x8x8
Conv3x3 > BatchNorm > Relu
1x256x8x8
1x256x8x8
Conv3x3
1x256x8x8
Add
1x256x8x8
BottleneckBlock1
1x256x8x8
1x256x8x8
BatchNorm > Relu
Conv1x1 > BatchNorm > Relu
Conv1x1
1x512x8x8
Conv3x3 > BatchNorm > Relu
1x2048x4x4
1x512x4x4
Conv1x1
1x2048x4x4
Add
BottleneckBlock2
1x2048x4x4
BatchNorm > Relu
1x2048x4x4
1x2048x4x4
Conv1x1 > BatchNorm > Relu
Conv1x1
1x512x4x4
Conv3x3 > BatchNorm > Relu
1x2048x4x4
1x512x4x4
Conv1x1
1x2048x4x4
Add
BottleneckBlock3
1x2048x4x4
BatchNorm > Relu
1x2048x4x4
1x2048x4x4
Conv1x1 > BatchNorm > Relu
Conv1x1
1x512x4x4
Conv3x3 > BatchNorm > Relu
1x2048x4x4
1x512x4x4
Conv1x1
1x2048x4x4
Add
1x2048x4x4
GlobalAveragePool
1x2048x1x1
FullyConnectedBlock
1x1024
FullyConnectedBlock