Segmentation d'image par architecture U-net

In [1]:
```
import os
import numpy as np
from PIL import Image
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPla
import time

%load_ext autoreload
%autoreload 2
```

Format des données

Les images sont toutes de 64x64 pixels, elles ont donc été redimensionnées pour faire la même taille.

D'autre part on constate qu'on a 3 canaux pour l'entrée, un seul en sortie. Ceci nous indique que les images initiales sont en couleur tandis que la sortie est constitué d'un canal binaire
(donc seulement le mask qui correspond à la classe de l'objet, on n'utilise pas le contour et le background?).

Enfin on a 738 images au total (~1/10 du dataset initial). On les sépare en 590 exemples d'entrainemene et 148 de test (soit un partitionnement de 80% / 20%).

In [2]:
```
X = np.load("data64_red.npy")
Y = np.load("mask64_red.npy")

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size

print("X shape:", X.shape,
      "\nY shape:", Y.shape,
      "\nX_train shape:", X_train.shape,
      "\nY_train shape:", Y_train.shape,
      "\nX_test shape:", X_test.shape,
      "\nY_test shape:", Y_test.shape)
```

```
X shape: (738, 64, 64, 3)
Y shape: (738, 64, 64, 1)
X_train shape: (590, 64, 64, 3)
Y_train shape: (590, 64, 64, 1)
X_test shape: (148, 64, 64, 3)
Y_test shape: (148, 64, 64, 1)
```

Réseau U-net simpliste

Comme on veut une profondeur de 128 au bottleneck, on utilise une conv2D de 128 à ce stade. La couche précédente et suivante doit être de même taille car on veut une symétrie entre la partie descendante et ascendante pour petmettre la concaténation, ici 64.

Puisque l'on travail avec une classe binaire, on utilise une fonction d'activation final en sigmoid (résultat entre 0 et 1). Pour la même raison, on choisit binary_crossentropy pour la fonction loss

In [3]:
```
from TP4_utils import unet_simple, affiche, display_results, eval_c
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStop
from tensorflow.keras import losses
import tensorflow as tf

bce = losses.BinaryCrossentropy()
dice = losses.Dice()
alpha = 0.5  # à optimiser éventuellement
def bce_dice_loss(y_true, y_pred):
    return alpha * bce(y_true, y_pred) + (1 - alpha) * dice(y_true,

def dice_coef(y_true, y_pred, smooth=1e-6):
    y_true_f = tf.reshape(y_true, [-1])
    y_pred_f = tf.reshape(y_pred, [-1])
    intersection = tf.reduce_sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (tf.reduce_sum(y_true_f)

lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, pa
earlStop = EarlyStopping(monitor='val_loss', min_delta=1e-6, patien

# Parameters
model = unet_simple(X_train.shape[1:])
model.summary()

# Compilation parameters
lr = 1e-3 # base learning rate
ad = Adam(learning_rate=lr)
l = bce_dice_loss #bce #dice #bce_dice_loss
m = ["accuracy", dice_coef]

# Compiler le modèle
model.compile(
    optimizer=ad,
    loss=l,
    metrics=m
```

```
    )

# Training parameters
epochs = 128 # max epochs
batch_size = 8
cb = [earlStop, lr_scheduler]

# Entraîner le modèle
tps1 = time.time()
history = model.fit(X_train,
     Y_train,
     epochs=epochs,
     batch_size=batch_size,
     callbacks=cb,
     validation_data=(X_test, Y_test)
     )
tps2 = time.time()

affiche(history)
preds = model.predict(X_test)
display_results(X_test, Y_test, preds)
eval_classif(Y_test, preds)
print("Temps d'entraînement : {:.2f} secondes".format(tps2 - tps1))
```

Model: "functional"

Layer (type)	Output Shape	Param #	Conne
input_layer (InputLayer)	(None, 64, 64, 3)	0	–
c1 (Conv2D)	(None, 64, 64, 16)	448	input
c1_2 (Conv2D)	(None, 64, 64, 16)	2,320	c1[0]
p1 (MaxPooling2D)	(None, 32, 32, 16)	0	c1_2[
c2 (Conv2D)	(None, 32, 32, 32)	4,640	p1[0]
c2_2 (Conv2D)	(None, 32, 32, 32)	9,248	c2[0]
p2 (MaxPooling2D)	(None, 16, 16, 32)	0	c2_2[
c3 (Conv2D)	(None, 16, 16, 64)	18,496	p2[0]
c3_2 (Conv2D)	(None, 16, 16, 64)	36,928	c3[0]
dropout (Dropout)	(None, 16, 16, 64)	0	c3_2[

p3 (MaxPooling2D)	(None, 8, 8, 64)	0	dropo
b (Conv2D)	(None, 8, 8, 128)	73,856	p3[0]
dropout_1 (Dropout)	(None, 8, 8, 128)	0	b[0][
u3 (Conv2DTranspose)	(None, 16, 16, 64)	73,792	dropo
u3b (Concatenate)	(None, 16, 16, 128)	0	u3[0] dropo
cd3 (Conv2D)	(None, 16, 16, 64)	73,792	u3b[0
cd3_2 (Conv2D)	(None, 16, 16, 64)	36,928	cd3[0
u2 (Conv2DTranspose)	(None, 32, 32, 32)	18,464	cd3_2
u2b (Concatenate)	(None, 32, 32, 64)	0	u2[0] c2_2[
cd2 (Conv2D)	(None, 32, 32, 32)	18,464	u2b[0
cd2_2 (Conv2D)	(None, 32, 32, 32)	9,248	cd2[0
u1 (Conv2DTranspose)	(None, 64, 64, 16)	4,624	cd2_2
u1b (Concatenate)	(None, 64, 64, 32)	0	u1[0] c1_2[
cd1 (Conv2D)	(None, 64, 64, 16)	4,624	u1b[0
cd1_2 (Conv2D)	(None, 64, 64, 16)	2,320	cd1[0
output1 (Conv2D)	(None, 64, 64, 1)	17	cd1_2

Total params: 388,209 (1.48 MB)

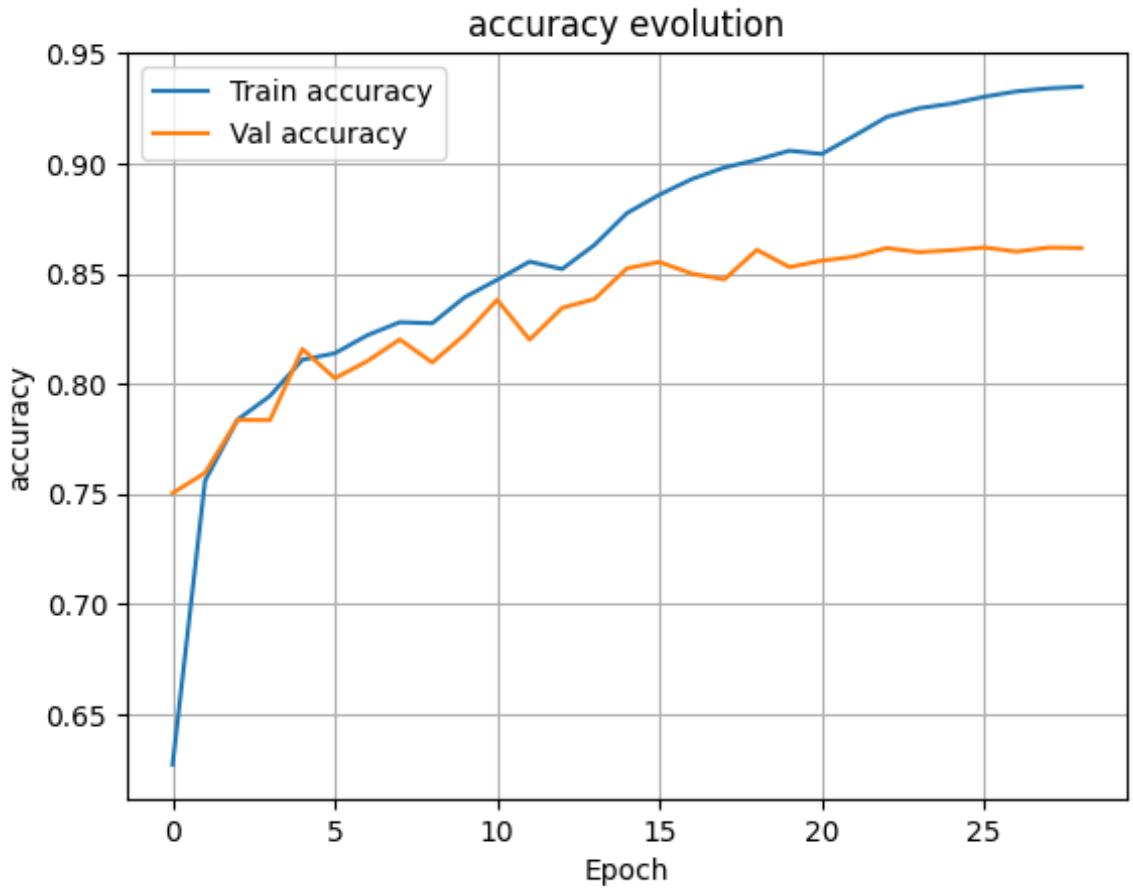
Trainable params: 388,209 (1.48 MB)

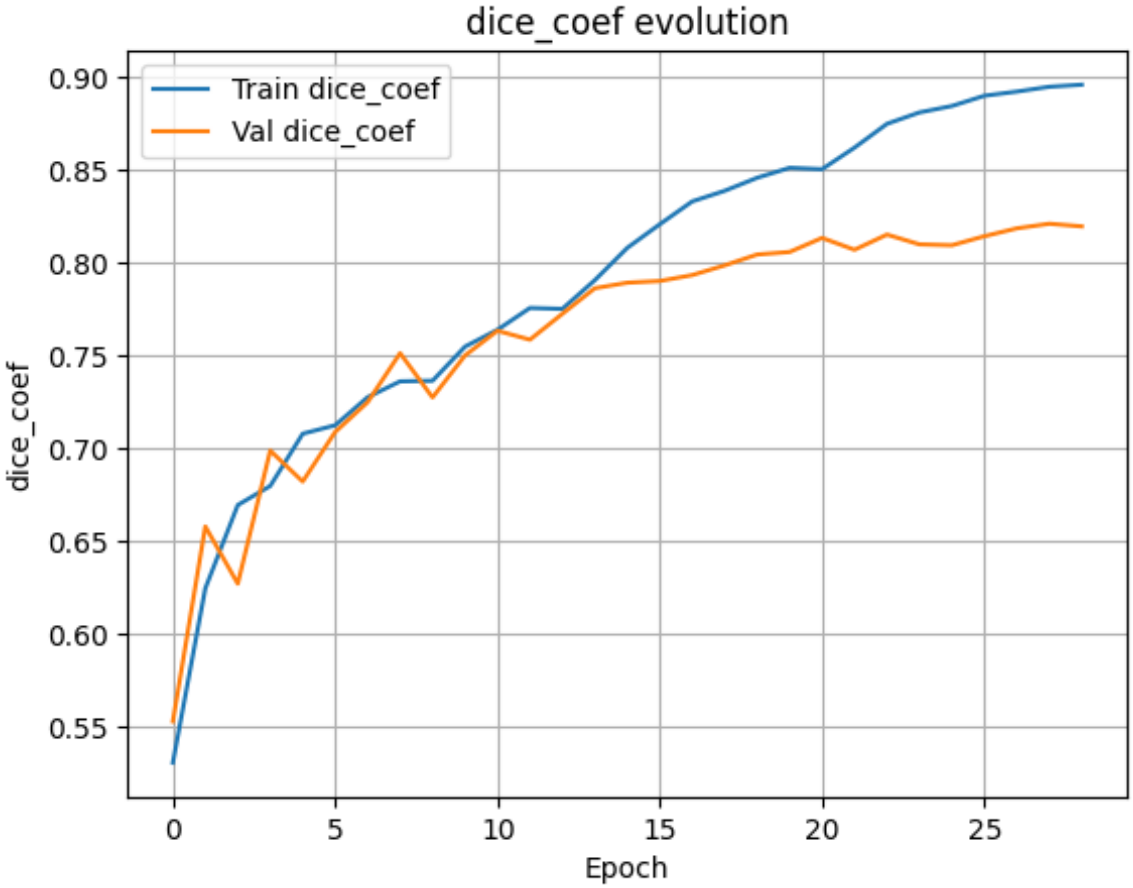
Non-trainable params: 0 (0.00 B)

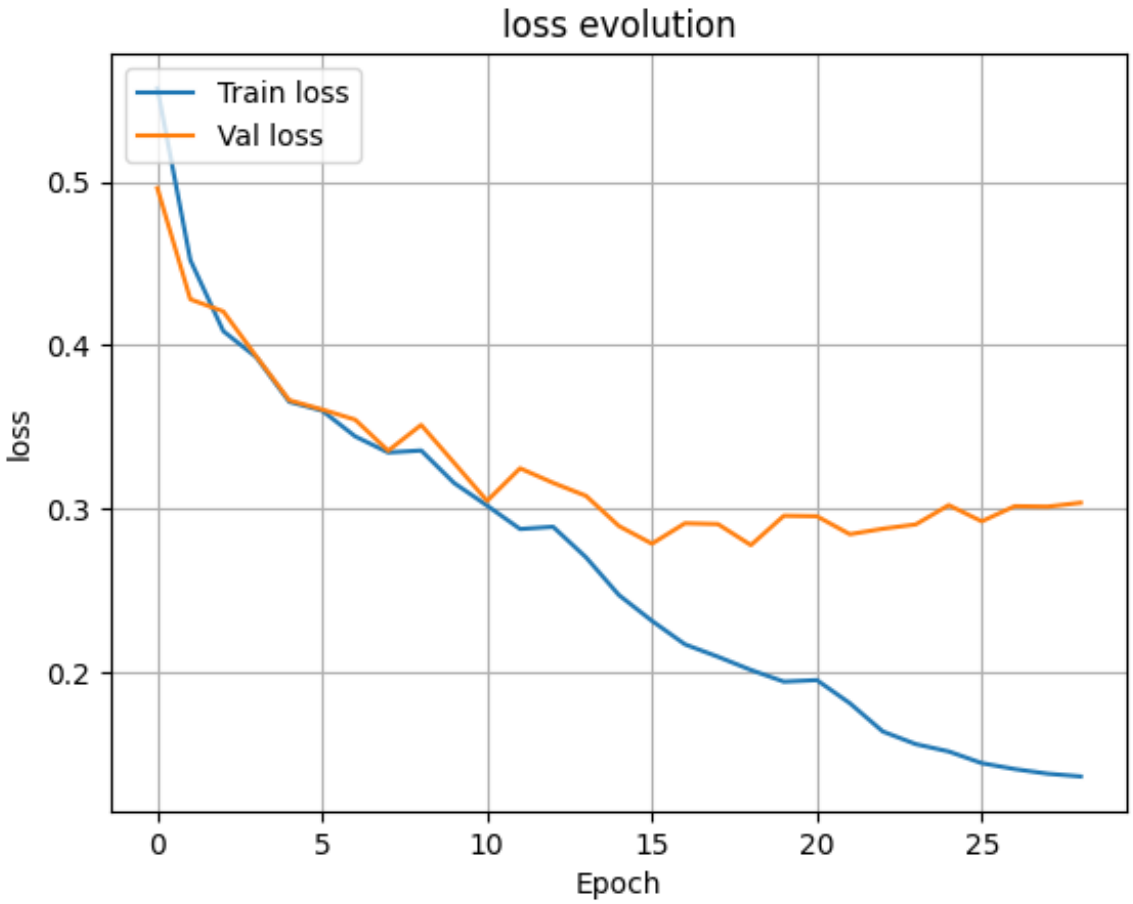
```
Epoch 1/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 6s 48ms/step - accuracy: 0.6270 - dice_co
ef: 0.5311 - loss: 0.5572 - val_accuracy: 0.7503 - val_dice_coef: 0.
5533 - val_loss: 0.4961 - learning_rate: 0.0010
Epoch 2/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.7559 - dice_co
ef: 0.6249 - loss: 0.4521 - val_accuracy: 0.7595 - val_dice_coef: 0.
6582 - val_loss: 0.4282 - learning_rate: 0.0010
Epoch 3/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.7836 - dice_co
```

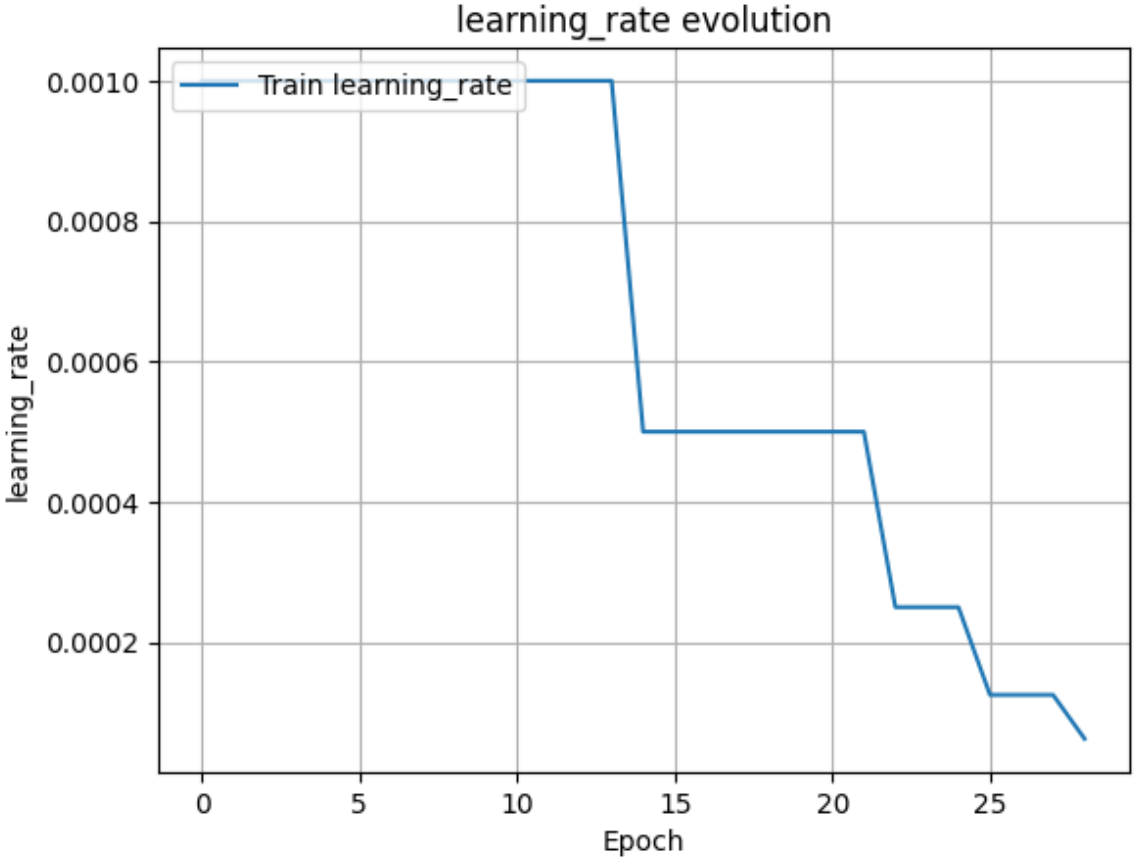
```
ef: 0.6697 - loss: 0.4087 - val_accuracy: 0.7836 - val_dice_coef: 0.
6274 - val_loss: 0.4207 - learning_rate: 0.0010
Epoch 4/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - accuracy: 0.7944 - dice_co
ef: 0.6800 - loss: 0.3928 - val_accuracy: 0.7834 - val_dice_coef: 0.
6990 - val_loss: 0.3933 - learning_rate: 0.0010
Epoch 5/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.8109 - dice_co
ef: 0.7081 - loss: 0.3655 - val_accuracy: 0.8157 - val_dice_coef: 0.
6824 - val_loss: 0.3665 - learning_rate: 0.0010
Epoch 6/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - accuracy: 0.8138 - dice_co
ef: 0.7127 - loss: 0.3601 - val_accuracy: 0.8025 - val_dice_coef: 0.
7091 - val_loss: 0.3608 - learning_rate: 0.0010
Epoch 7/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.8220 - dice_co
ef: 0.7278 - loss: 0.3444 - val_accuracy: 0.8103 - val_dice_coef: 0.
7251 - val_loss: 0.3546 - learning_rate: 0.0010
Epoch 8/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - accuracy: 0.8279 - dice_co
ef: 0.7363 - loss: 0.3346 - val_accuracy: 0.8201 - val_dice_coef: 0.
7515 - val_loss: 0.3356 - learning_rate: 0.0010
Epoch 9/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.8274 - dice_co
ef: 0.7366 - loss: 0.3358 - val_accuracy: 0.8096 - val_dice_coef: 0.
7277 - val_loss: 0.3514 - learning_rate: 0.0010
Epoch 10/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 40ms/step - accuracy: 0.8393 - dice_co
ef: 0.7550 - loss: 0.3159 - val_accuracy: 0.8223 - val_dice_coef: 0.
7503 - val_loss: 0.3282 - learning_rate: 0.0010
Epoch 11/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 4s 56ms/step - accuracy: 0.8472 - dice_co
ef: 0.7641 - loss: 0.3021 - val_accuracy: 0.8381 - val_dice_coef: 0.
7636 - val_loss: 0.3050 - learning_rate: 0.0010
Epoch 12/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.8554 - dice_co
ef: 0.7758 - loss: 0.2878 - val_accuracy: 0.8200 - val_dice_coef: 0.
7588 - val_loss: 0.3249 - learning_rate: 0.0010
Epoch 13/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - accuracy: 0.8520 - dice_co
ef: 0.7753 - loss: 0.2891 - val_accuracy: 0.8344 - val_dice_coef: 0.
7727 - val_loss: 0.3160 - learning_rate: 0.0010
Epoch 14/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.8630 - dice_co
ef: 0.7908 - loss: 0.2705 - val_accuracy: 0.8384 - val_dice_coef: 0.
7865 - val_loss: 0.3080 - learning_rate: 0.0010
Epoch 15/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 39ms/step - accuracy: 0.8775 - dice_co
ef: 0.8082 - loss: 0.2474 - val_accuracy: 0.8524 - val_dice_coef: 0.
7894 - val_loss: 0.2896 - learning_rate: 5.0000e-04
Epoch 16/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 40ms/step - accuracy: 0.8858 - dice_co
ef: 0.8209 - loss: 0.2317 - val_accuracy: 0.8553 - val_dice_coef: 0.
7903 - val_loss: 0.2788 - learning_rate: 5.0000e-04
Epoch 17/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 40ms/step - accuracy: 0.8929 - dice_co
```

```
ef: 0.8332 - loss: 0.2173 - val_accuracy: 0.8499 - val_dice_coef: 0.
7935 - val_loss: 0.2913 - learning_rate: 5.0000e-04
Epoch 18/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.8981 - dice_co
ef: 0.8389 - loss: 0.2097 - val_accuracy: 0.8473 - val_dice_coef: 0.
7988 - val_loss: 0.2908 - learning_rate: 5.0000e-04
Epoch 19/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - accuracy: 0.9016 - dice_co
ef: 0.8459 - loss: 0.2017 - val_accuracy: 0.8608 - val_dice_coef: 0.
8046 - val_loss: 0.2780 - learning_rate: 5.0000e-04
Epoch 20/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9058 - dice_co
ef: 0.8513 - loss: 0.1945 - val_accuracy: 0.8530 - val_dice_coef: 0.
8060 - val_loss: 0.2959 - learning_rate: 5.0000e-04
Epoch 21/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - accuracy: 0.9043 - dice_co
ef: 0.8505 - loss: 0.1955 - val_accuracy: 0.8558 - val_dice_coef: 0.
8135 - val_loss: 0.2956 - learning_rate: 5.0000e-04
Epoch 22/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - accuracy: 0.9126 - dice_co
ef: 0.8621 - loss: 0.1813 - val_accuracy: 0.8577 - val_dice_coef: 0.
8072 - val_loss: 0.2846 - learning_rate: 5.0000e-04
Epoch 23/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - accuracy: 0.9211 - dice_co
ef: 0.8750 - loss: 0.1641 - val_accuracy: 0.8616 - val_dice_coef: 0.
8153 - val_loss: 0.2880 - learning_rate: 2.5000e-04
Epoch 24/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 46ms/step - accuracy: 0.9251 - dice_co
ef: 0.8811 - loss: 0.1563 - val_accuracy: 0.8597 - val_dice_coef: 0.
8101 - val_loss: 0.2906 - learning_rate: 2.5000e-04
Epoch 25/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.9272 - dice_co
ef: 0.8846 - loss: 0.1518 - val_accuracy: 0.8606 - val_dice_coef: 0.
8096 - val_loss: 0.3024 - learning_rate: 2.5000e-04
Epoch 26/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.9303 - dice_co
ef: 0.8901 - loss: 0.1447 - val_accuracy: 0.8619 - val_dice_coef: 0.
8144 - val_loss: 0.2925 - learning_rate: 1.2500e-04
Epoch 27/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - accuracy: 0.9327 - dice_co
ef: 0.8923 - loss: 0.1411 - val_accuracy: 0.8600 - val_dice_coef: 0.
8187 - val_loss: 0.3017 - learning_rate: 1.2500e-04
Epoch 28/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - accuracy: 0.9341 - dice_co
ef: 0.8949 - loss: 0.1383 - val_accuracy: 0.8618 - val_dice_coef: 0.
8212 - val_loss: 0.3015 - learning_rate: 1.2500e-04
Epoch 29/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - accuracy: 0.9349 - dice_co
ef: 0.8960 - loss: 0.1366 - val_accuracy: 0.8616 - val_dice_coef: 0.
8198 - val_loss: 0.3039 - learning_rate: 6.2500e-05
Epoch 29: early stopping
Restoring model weights from the end of the best epoch: 19.
```


accuracy evolution
Train accuracy
Val accuracy
accuracy
Epoch
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
5
10
15
20
25


dice_coef evolution
Train dice_coef
Val dice_coef
dice_coef
Epoch
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
5
10
15
20
25





5/5 ━━━━━━━━━━━━━━━━━━━━ **0s** 53ms/step

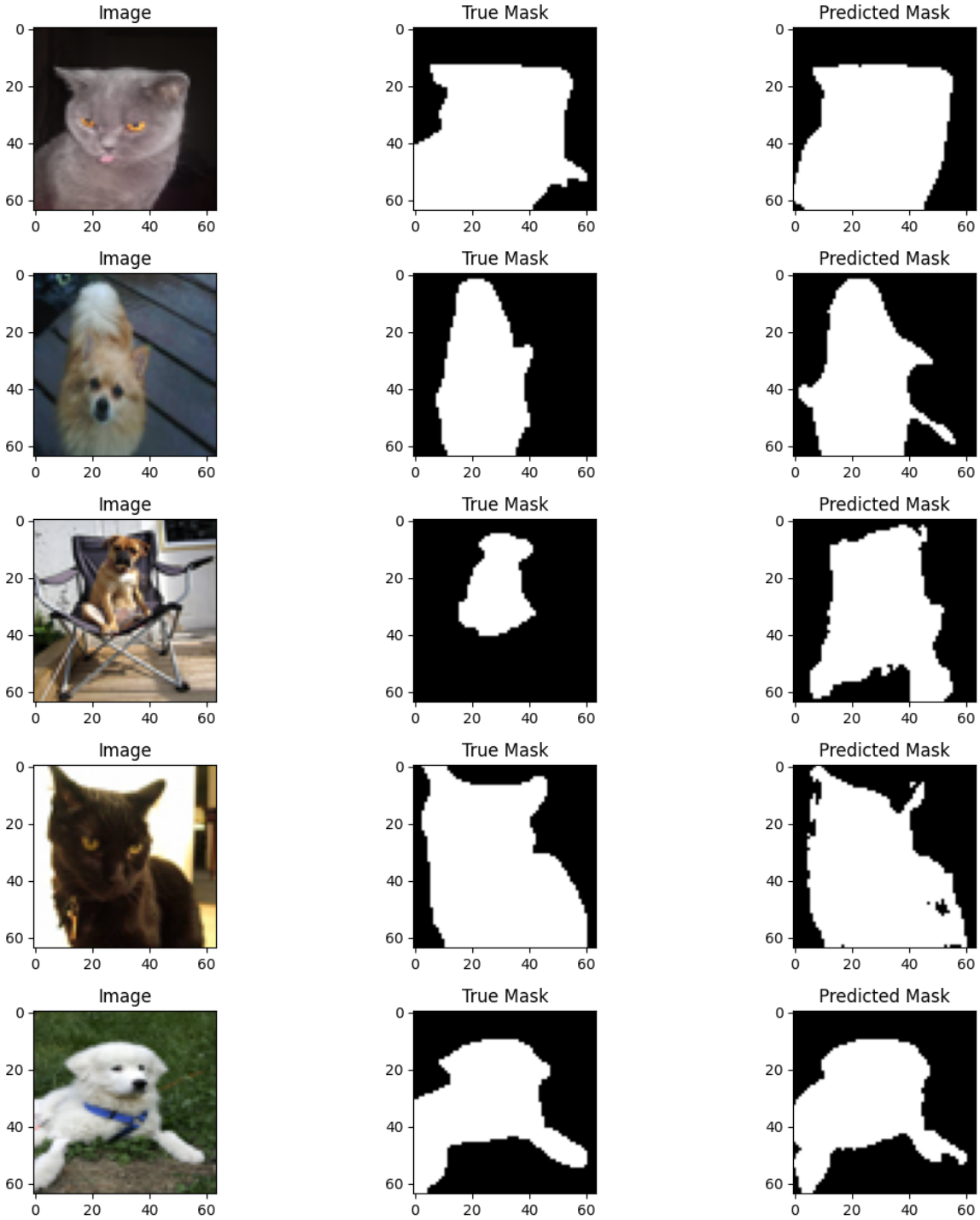
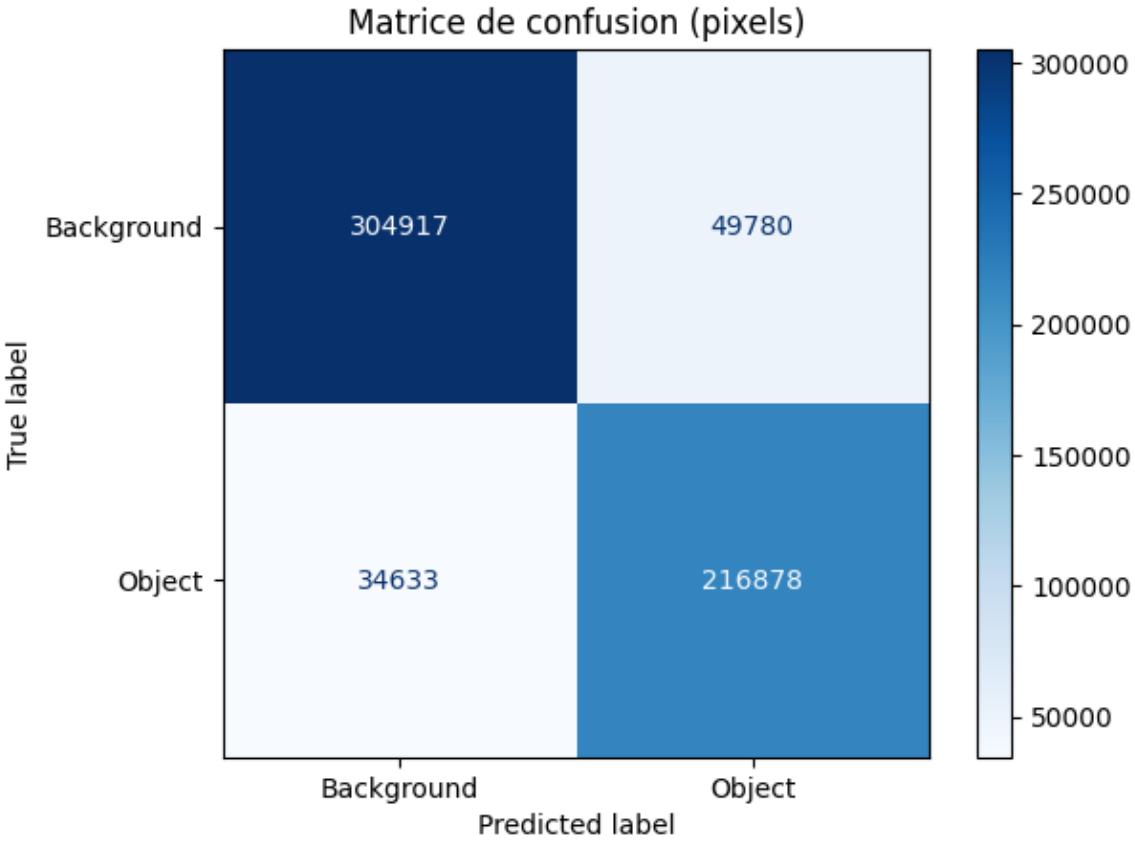


Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask




```
Classification report (par pixel):
              precision    recall  f1-score   support

  Background       0.90      0.86      0.88    354697
      Object       0.81      0.86      0.84    251511

    accuracy                           0.86    606208
   macro avg       0.86      0.86      0.86    606208
weighted avg       0.86      0.86      0.86    606208


Précision globale (pixel accuracy): 0.8608
Temps d'entraînement : 95.86 secondes
```

1 - profondeur de la structure

On commence par s'interesser au nombre de niveau de la structure. À chaque niveau de plus, on ajoute un couple maxpooling / Conv2DTranspose dans l'encodeur et décodeur. On conserve donc un facteur 2 pour les filtres entre chaque niveau. Cela donne par exemple pour 4 niveaux : 16 - 32 - 64 - 128 (bottleneck).

itération	fct perte	profondeur	epoch	temps (s)	nb params	accuracy
1	`bce`	1	10	58.73	182,337	**73.76%**
2	`bce`	2	10	36.20	226,593	**77.38%**
3	`bce`	3	10	19.82	237,457	**78.25%**
4	`bce`	4	10	12.88	240,073	**76.19%**

5	`bce`	5	10	10.69	240,677	**74.16%**

On peut déjà faire plusieurs observations :

- Le temps d'entraînement diminue avec le nombre de couches. Ce phénomène est d'abord contre-intuitif car le nombre de paramètres augmente. Cependant, ceux-ci sont plus répartis, et la structure tire avantage des max-pooling en réduisant la dimension en parallèle de l'augmentation du nombre de filtres. Ainsi, les couches sont globalement plus petites, ce qui limite le temps de calcul.
- Le gain en nombre de paramètres est de moins en moins significatif, donc l'enrichissement du réseau entre 3, 4 et 5 couches est limité.
- La précision augmente sur les trois premiers niveaux puis stagne, voire diminue sur les deux derniers. De plus, on commence à observer des signes d'overfitting sur ceux-ci.

On gardera donc 3 niveaux par la suite.

2 - Couches de convolutions par niveau d'encodeur/décodeur

Pour augmenter la capacité du réseau, on peut alors augmenter le nombre de convolutions par étage :

itération	fct perte	nb couches	epoch	temps (s)	nb params	accuracy
0	`bce`	1	10	19.82	237,457	**78.25%**
1	`bce`	2	10	31.44	334,449	**78.89%**
2	`bce`	3	10	42.92	431,441	**77.52%**

La précision est en effet améliorée avec 2 couches, bien que ce ne soit pas de beaucoup.
Dès la troisième couche, elle est dégradée (probablement signe d'overfitting) et le temps d'apprentissage est nettement plus long.
Par la suite, on utilisera 2 couches par niveau.

3 - UpSampling2D ou Conv2DTranspose

Ce choix nous intéresse car Conv2DTranspose fait intervenir des paramètres, contrairement à UpSampling2D. Ainsi, analyser les résultats obtenus entre les deux permet de vérifier si ces paramètres sont réellement expressifs. Dans le cas contraire, UpSampling2D serait préférable car il réduit le temps d'apprentissage.

itération	fct perte	méthode	epoch	temps (s)	nb params	accuracy
0	`bce`	Conv2DTranspose	10	31.44	334,449	**78.89%**

1	`bce`	UpSampling2D	10	36.10	339,713	**79.06%**

Bien qu'UpSampling2D introduise en fait légèrement plus de paramètres (effet du couplage avec les convolutions suivantes), les performances sont légèrement meilleures. On choisit UpSampling2D pour la suite.

4 - Nombre de filtres au bottleneck

On ajuste le nombre de filtres à l'encodeur et au décodeur en conséquence.

itération	fct perte	nb filtres	epoch	temps (s)	nb params	accuracy
0	`bce`	128	10	36.10	339,713	**79.06%**
1	`bce`	64	10	18.44	85,185	**77.84%**
2	`bce`	256	10	94.10	1,356,801	**81.29%**

Comme on peut s'y attendre, un bottleneck plus grand permet au réseau de mieux s'exprimer et la précision augmente.
Cependant, un bottleneck à 256 filtres fait exploser le nombre de paramètres et le temps d'apprentissage.
Nous resterons avec 128 filtres au bottleneck par la suite.

5 - Fonction de perte

Pour cette partie, on ajoute `dice_coef` (1 - Dice) comme métrique pour pouvoir évaluer le résultat.

itération	fct perte	epoch	temps (s)	dice_coef	accuracy
0	`bce`	10	36.10	**64.02%**	**79.17%**
1	`dice`	10	35.85	**74.20%**	**76.28%**
2	`bce + dice`	10	36.29	**69.07%**	**80.28%**

`bce + dice` donne le meilleur compromis.

Quelques ajouts

Pour aller plus loin, on peut ajouter un **early stop** pour optimiser le nombre d'epochs, puis un **lr scheduler** (reduce on plateau) pour accélérer la convergence au début, tout en permettant d'affiner celle-ci sur la fin, ainsi qu'un **dropout** au bottleneck pour réduire l'overfitting.
J'obtiens alors des résultats tels que :

```
Classification report (par pixel):
              precision    recall  f1-score   support

  Background       0.88      0.89      0.89    354697
```

```
      Object       0.85      0.83      0.84    251511

    accuracy                           0.87    606208
   macro avg       0.87      0.86      0.87    606208
weighted avg       0.87      0.87      0.87    606208

Précision globale (pixel accuracy): 0.8698
Temps d'entraînement : 99.45 secondes
```

avec dice_coef > 82%

In [11]:

```python
from TP4_utils import unet_deep, affiche_deep

def prepare_sortie(Y):
    y1 = Y
    y2 = tf.image.resize(Y, (32, 32), method='nearest')
    y3 = tf.image.resize(Y, (16, 16), method='nearest')
    return [y1, y2, y3]

Y_train_new = prepare_sortie(Y_train)
Y_test_new = prepare_sortie(Y_test)

model_deep = unet_deep(X_train.shape[1:])
model_deep.summary()

lr_scheduler_deep = ReduceLROnPlateau(
    monitor='val_output1_loss',
    factor=0.5,
    patience=3,
    min_lr=1e-6,
    mode='min'
)
earlStop_deep = EarlyStopping(
    monitor='val_output1_loss',
    min_delta=1e-4,
    patience=15,
    restore_best_weights=True,
    verbose=1,
    mode='min'
)
cb_deep = [earlStop_deep, lr_scheduler_deep]

model_deep.compile(
    optimizer="adam",
    loss={
        "output1": bce_dice_loss,
        "output2": bce_dice_loss,
        "output3": bce_dice_loss
    },
    loss_weights={
        "output1": 1,
        "output2": 0.5,
        "output3": 0.25
    },
    metrics={
```

```
        "output1": ["accuracy", dice_coef],
        "output2": ["accuracy", dice_coef],
        "output3": ["accuracy", dice_coef]
    }
)

tps1 = time.time()
history = model_deep.fit(X_train,
     Y_train_new,
     epochs=epochs,
     batch_size=batch_size,
     callbacks=cb_deep,
     validation_data=(X_test, Y_test_new)
     )
tps2 = time.time()

affiche_deep(history)
preds = model_deep.predict(X_test)
display_results(X_test, Y_test, preds[0])
eval_classif(Y_test, preds[0])
print("Temps d'entraînement : {:.2f} secondes".format(tps2 - tps1))
```

Model: "functional_8"

Layer (type)	Output Shape	Param #	Conne
input_layer_8 (InputLayer)	(None, 64, 64, 3)	0	–
c1 (Conv2D)	(None, 64, 64, 16)	448	input
c1_2 (Conv2D)	(None, 64, 64, 16)	2,320	c1[0]
p1 (MaxPooling2D)	(None, 32, 32, 16)	0	c1_2[
c2 (Conv2D)	(None, 32, 32, 32)	4,640	p1[0]
c2_2 (Conv2D)	(None, 32, 32, 32)	9,248	c2[0]
p2 (MaxPooling2D)	(None, 16, 16, 32)	0	c2_2[
c3 (Conv2D)	(None, 16, 16, 64)	18,496	p2[0]
c3_2 (Conv2D)	(None, 16, 16, 64)	36,928	c3[0]
dropout_16 (Dropout)	(None, 16, 16, 64)	0	c3_2[
p3 (MaxPooling2D)	(None, 8, 8, 64)	0	dropo

b (Conv2D)	(None, 8, 8, 128)	73,856	p3[0]
dropout_17 (Dropout)	(None, 8, 8, 128)	0	b[0][
u3 (Conv2DTranspose)	(None, 16, 16, 64)	73,792	dropo
u3b (Concatenate)	(None, 16, 16, 128)	0	u3[0] dropo
cd3 (Conv2D)	(None, 16, 16, 64)	73,792	u3b[0
cd3_2 (Conv2D)	(None, 16, 16, 64)	36,928	cd3[0
u2 (Conv2DTranspose)	(None, 32, 32, 32)	18,464	cd3_2
u2b (Concatenate)	(None, 32, 32, 64)	0	u2[0] c2_2[
cd2 (Conv2D)	(None, 32, 32, 32)	18,464	u2b[0
cd2_2 (Conv2D)	(None, 32, 32, 32)	9,248	cd2[0
u1 (Conv2DTranspose)	(None, 64, 64, 16)	4,624	cd2_2
u1b (Concatenate)	(None, 64, 64, 32)	0	u1[0] c1_2[
cd1 (Conv2D)	(None, 64, 64, 16)	4,624	u1b[0
cd1_2 (Conv2D)	(None, 64, 64, 16)	2,320	cd1[0
output1 (Conv2D)	(None, 64, 64, 1)	17	cd1_2
output2 (Conv2D)	(None, 32, 32, 1)	33	cd2_2
output3 (Conv2D)	(None, 16, 16, 1)	65	cd3_2

Total params: 388,307 (1.48 MB)

Trainable params: 388,307 (1.48 MB)

Non-trainable params: 0 (0.00 B)

Epoch 1/128

74/74 ━━━━━━━━━━━━━━━━━━━━ **6s** 45ms/step - loss: 0.9089 - output1_acc uracy: 0.6887 - output1_dice_coef: 0.5673 - output1_loss: 0.5180 - o utput2_accuracy: 0.6868 - output2_dice_coef: 0.5660 - output2_loss: 0.5206 - output3_accuracy: 0.6832 - output3_dice_coef: 0.5765 - outp ut3_loss: 0.5201 - val_loss: 0.8088 - val_output1_accuracy: 0.7282 - val_output1_dice_coef: 0.6506 - val_output1_loss: 0.4596 - val_outpu

```
t2_accuracy: 0.7257 - val_output2_dice_coef: 0.6403 - val_output2_lo
ss: 0.4601 - val_output3_accuracy: 0.7236 - val_output3_dice_coef:
0.6278 - val_output3_loss: 0.4737 - learning_rate: 0.0010
Epoch 2/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.8077 - output1_acc
uracy: 0.7480 - output1_dice_coef: 0.6202 - output1_loss: 0.4600 - o
utput2_accuracy: 0.7449 - output2_dice_coef: 0.6194 - output2_loss:
0.4617 - output3_accuracy: 0.7443 - output3_dice_coef: 0.6140 - outp
ut3_loss: 0.4683 - val_loss: 0.8212 - val_output1_accuracy: 0.7727 -
val_output1_dice_coef: 0.5623 - val_output1_loss: 0.4675 - val_outpu
t2_accuracy: 0.7712 - val_output2_dice_coef: 0.5683 - val_output2_lo
ss: 0.4657 - val_output3_accuracy: 0.7543 - val_output3_dice_coef:
0.5562 - val_output3_loss: 0.4848 - learning_rate: 0.0010
Epoch 3/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.7415 - output1_acc
uracy: 0.7698 - output1_dice_coef: 0.6570 - output1_loss: 0.4211 - o
utput2_accuracy: 0.7685 - output2_dice_coef: 0.6510 - output2_loss:
0.4242 - output3_accuracy: 0.7662 - output3_dice_coef: 0.6423 - outp
ut3_loss: 0.4332 - val_loss: 0.7673 - val_output1_accuracy: 0.7302 -
val_output1_dice_coef: 0.6731 - val_output1_loss: 0.4429 - val_outpu
t2_accuracy: 0.7406 - val_output2_dice_coef: 0.6691 - val_output2_lo
ss: 0.4340 - val_output3_accuracy: 0.7582 - val_output3_dice_coef:
0.6595 - val_output3_loss: 0.4317 - learning_rate: 0.0010
Epoch 4/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.7255 - output1_acc
uracy: 0.7769 - output1_dice_coef: 0.6640 - output1_loss: 0.4138 - o
utput2_accuracy: 0.7755 - output2_dice_coef: 0.6612 - output2_loss:
0.4151 - output3_accuracy: 0.7751 - output3_dice_coef: 0.6557 - outp
ut3_loss: 0.4195 - val_loss: 0.6830 - val_output1_accuracy: 0.7995 -
val_output1_dice_coef: 0.6611 - val_output1_loss: 0.3873 - val_outpu
t2_accuracy: 0.7965 - val_output2_dice_coef: 0.6612 - val_output2_lo
ss: 0.3892 - val_output3_accuracy: 0.7835 - val_output3_dice_coef:
0.6485 - val_output3_loss: 0.4103 - learning_rate: 0.0010
Epoch 5/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.6753 - output1_acc
uracy: 0.7963 - output1_dice_coef: 0.6913 - output1_loss: 0.3834 - o
utput2_accuracy: 0.7944 - output2_dice_coef: 0.6863 - output2_loss:
0.3862 - output3_accuracy: 0.7914 - output3_dice_coef: 0.6766 - outp
ut3_loss: 0.3942 - val_loss: 0.6437 - val_output1_accuracy: 0.8086 -
val_output1_dice_coef: 0.6958 - val_output1_loss: 0.3652 - val_outpu
t2_accuracy: 0.8079 - val_output2_dice_coef: 0.6882 - val_output2_lo
ss: 0.3697 - val_output3_accuracy: 0.8023 - val_output3_dice_coef:
0.6783 - val_output3_loss: 0.3817 - learning_rate: 0.0010
Epoch 6/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.6537 - output1_acc
uracy: 0.8061 - output1_dice_coef: 0.7025 - output1_loss: 0.3712 - o
utput2_accuracy: 0.8033 - output2_dice_coef: 0.6971 - output2_loss:
0.3737 - output3_accuracy: 0.8002 - output3_dice_coef: 0.6855 - outp
ut3_loss: 0.3805 - val_loss: 0.6381 - val_output1_accuracy: 0.8104 -
val_output1_dice_coef: 0.6974 - val_output1_loss: 0.3630 - val_outpu
t2_accuracy: 0.8087 - val_output2_dice_coef: 0.6953 - val_output2_lo
ss: 0.3656 - val_output3_accuracy: 0.8037 - val_output3_dice_coef:
0.6824 - val_output3_loss: 0.3769 - learning_rate: 0.0010
Epoch 7/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.6151 - output1_acc
uracy: 0.8182 - output1_dice_coef: 0.7207 - output1_loss: 0.3493 - o
```

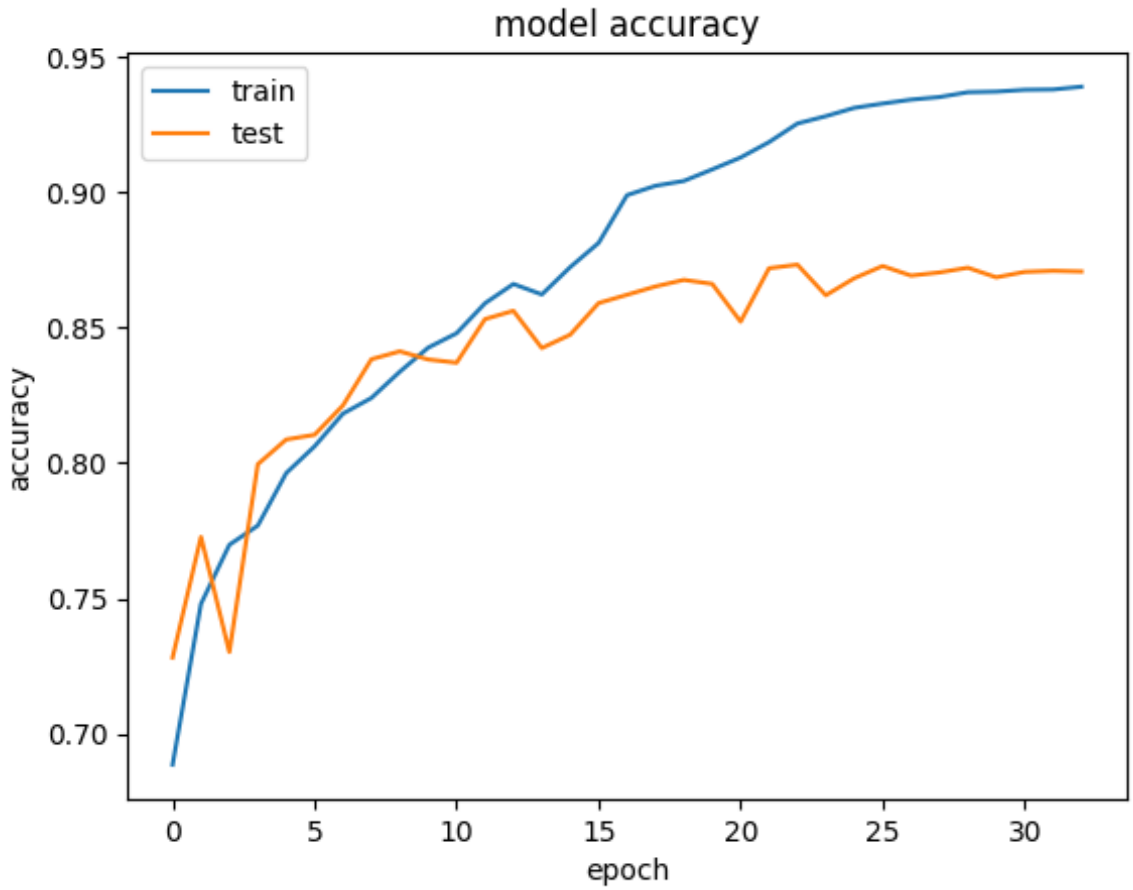
```
utput2_accuracy: 0.8166 - output2_dice_coef: 0.7156 - output2_loss:
0.3516 - output3_accuracy: 0.8119 - output3_dice_coef: 0.7025 - outp
ut3_loss: 0.3593 - val_loss: 0.6032 - val_output1_accuracy: 0.8212 -
val_output1_dice_coef: 0.7393 - val_output1_loss: 0.3416 - val_outpu
t2_accuracy: 0.8187 - val_output2_dice_coef: 0.7355 - val_output2_lo
ss: 0.3471 - val_output3_accuracy: 0.8080 - val_output3_dice_coef:
0.7191 - val_output3_loss: 0.3633 - learning_rate: 0.0010
Epoch 8/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - loss: 0.5949 - output1_acc
uracy: 0.8240 - output1_dice_coef: 0.7331 - output1_loss: 0.3379 - o
utput2_accuracy: 0.8224 - output2_dice_coef: 0.7288 - output2_loss:
0.3400 - output3_accuracy: 0.8153 - output3_dice_coef: 0.7147 - outp
ut3_loss: 0.3506 - val_loss: 0.5563 - val_output1_accuracy: 0.8383 -
val_output1_dice_coef: 0.7508 - val_output1_loss: 0.3150 - val_outpu
t2_accuracy: 0.8356 - val_output2_dice_coef: 0.7459 - val_output2_lo
ss: 0.3191 - val_output3_accuracy: 0.8186 - val_output3_dice_coef:
0.7252 - val_output3_loss: 0.3467 - learning_rate: 0.0010
Epoch 9/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - loss: 0.5676 - output1_acc
uracy: 0.8337 - output1_dice_coef: 0.7456 - output1_loss: 0.3218 - o
utput2_accuracy: 0.8313 - output2_dice_coef: 0.7405 - output2_loss:
0.3242 - output3_accuracy: 0.8265 - output3_dice_coef: 0.7258 - outp
ut3_loss: 0.3350 - val_loss: 0.5467 - val_output1_accuracy: 0.8412 -
val_output1_dice_coef: 0.7753 - val_output1_loss: 0.3123 - val_outpu
t2_accuracy: 0.8397 - val_output2_dice_coef: 0.7710 - val_output2_lo
ss: 0.3168 - val_output3_accuracy: 0.8320 - val_output3_dice_coef:
0.7469 - val_output3_loss: 0.3282 - learning_rate: 0.0010
Epoch 10/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.5466 - output1_acc
uracy: 0.8426 - output1_dice_coef: 0.7566 - output1_loss: 0.3099 - o
utput2_accuracy: 0.8399 - output2_dice_coef: 0.7529 - output2_loss:
0.3124 - output3_accuracy: 0.8340 - output3_dice_coef: 0.7392 - outp
ut3_loss: 0.3228 - val_loss: 0.5421 - val_output1_accuracy: 0.8382 -
val_output1_dice_coef: 0.7731 - val_output1_loss: 0.3109 - val_outpu
t2_accuracy: 0.8394 - val_output2_dice_coef: 0.7693 - val_output2_lo
ss: 0.3099 - val_output3_accuracy: 0.8295 - val_output3_dice_coef:
0.7456 - val_output3_loss: 0.3288 - learning_rate: 0.0010
Epoch 11/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.5319 - output1_acc
uracy: 0.8479 - output1_dice_coef: 0.7637 - output1_loss: 0.3023 - o
utput2_accuracy: 0.8467 - output2_dice_coef: 0.7615 - output2_loss:
0.3025 - output3_accuracy: 0.8406 - output3_dice_coef: 0.7495 - outp
ut3_loss: 0.3120 - val_loss: 0.5418 - val_output1_accuracy: 0.8370 -
val_output1_dice_coef: 0.7652 - val_output1_loss: 0.3127 - val_outpu
t2_accuracy: 0.8385 - val_output2_dice_coef: 0.7586 - val_output2_lo
ss: 0.3119 - val_output3_accuracy: 0.8351 - val_output3_dice_coef:
0.7443 - val_output3_loss: 0.3217 - learning_rate: 0.0010
Epoch 12/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - loss: 0.4972 - output1_acc
uracy: 0.8589 - output1_dice_coef: 0.7818 - output1_loss: 0.2814 - o
utput2_accuracy: 0.8575 - output2_dice_coef: 0.7780 - output2_loss:
0.2831 - output3_accuracy: 0.8513 - output3_dice_coef: 0.7632 - outp
ut3_loss: 0.2940 - val_loss: 0.5012 - val_output1_accuracy: 0.8531 -
val_output1_dice_coef: 0.7887 - val_output1_loss: 0.2878 - val_outpu
t2_accuracy: 0.8521 - val_output2_dice_coef: 0.7831 - val_output2_lo
ss: 0.2890 - val_output3_accuracy: 0.8461 - val_output3_dice_coef:
```

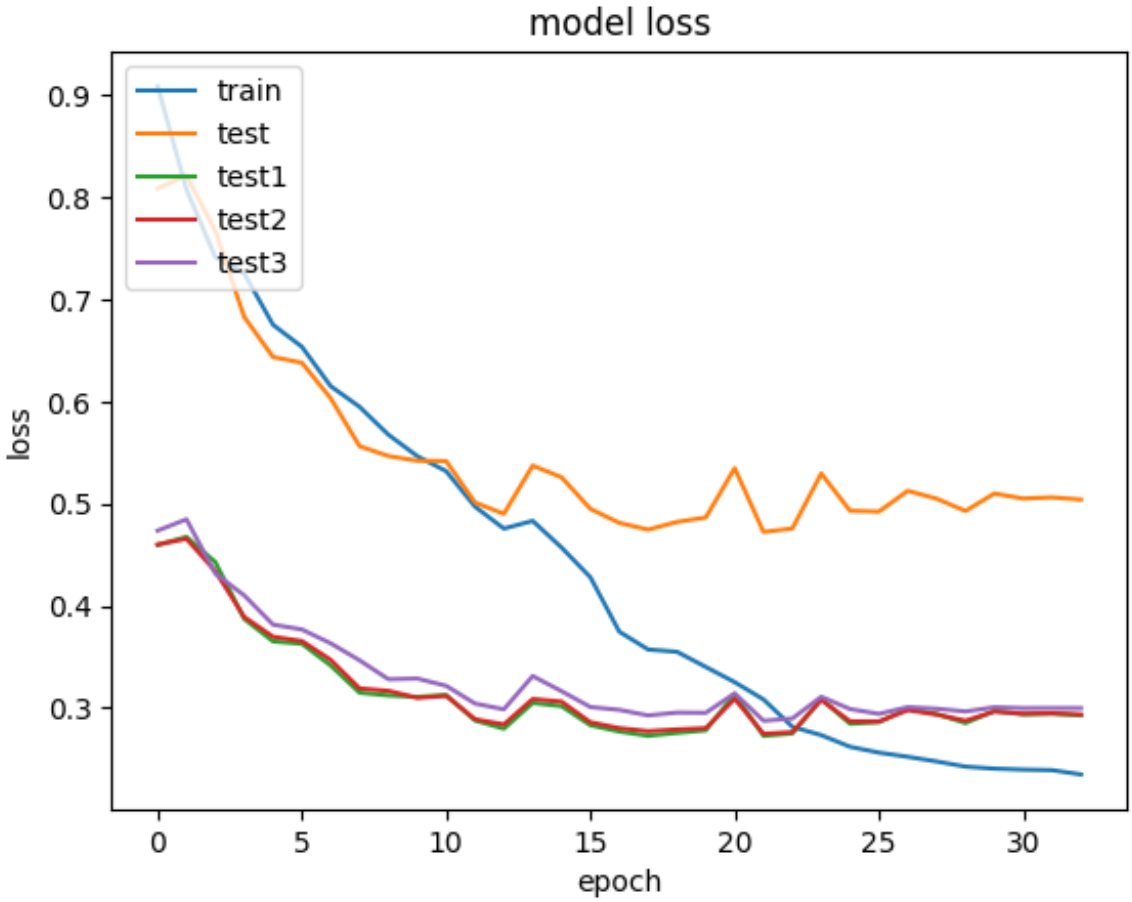
```
0.7644 - val_output3_loss: 0.3043 - learning_rate: 0.0010
Epoch 13/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.4757 - output1_acc
uracy: 0.8660 - output1_dice_coef: 0.7921 - output1_loss: 0.2696 - o
utput2_accuracy: 0.8641 - output2_dice_coef: 0.7897 - output2_loss:
0.2710 - output3_accuracy: 0.8573 - output3_dice_coef: 0.7766 - outp
ut3_loss: 0.2822 - val_loss: 0.4901 - val_output1_accuracy: 0.8561 -
val_output1_dice_coef: 0.7924 - val_output1_loss: 0.2797 - val_outpu
t2_accuracy: 0.8524 - val_output2_dice_coef: 0.7871 - val_output2_lo
ss: 0.2839 - val_output3_accuracy: 0.8458 - val_output3_dice_coef:
0.7681 - val_output3_loss: 0.2987 - learning_rate: 0.0010
Epoch 14/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.4832 - output1_acc
uracy: 0.8622 - output1_dice_coef: 0.7886 - output1_loss: 0.2739 - o
utput2_accuracy: 0.8602 - output2_dice_coef: 0.7865 - output2_loss:
0.2755 - output3_accuracy: 0.8536 - output3_dice_coef: 0.7741 - outp
ut3_loss: 0.2863 - val_loss: 0.5374 - val_output1_accuracy: 0.8424 -
val_output1_dice_coef: 0.7623 - val_output1_loss: 0.3049 - val_outpu
t2_accuracy: 0.8386 - val_output2_dice_coef: 0.7637 - val_output2_lo
ss: 0.3088 - val_output3_accuracy: 0.8275 - val_output3_dice_coef:
0.7515 - val_output3_loss: 0.3313 - learning_rate: 0.0010
Epoch 15/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - loss: 0.4571 - output1_acc
uracy: 0.8723 - output1_dice_coef: 0.8003 - output1_loss: 0.2584 - o
utput2_accuracy: 0.8702 - output2_dice_coef: 0.7972 - output2_loss:
0.2601 - output3_accuracy: 0.8627 - output3_dice_coef: 0.7805 - outp
ut3_loss: 0.2748 - val_loss: 0.5259 - val_output1_accuracy: 0.8473 -
val_output1_dice_coef: 0.7925 - val_output1_loss: 0.3018 - val_outpu
t2_accuracy: 0.8427 - val_output2_dice_coef: 0.7883 - val_output2_lo
ss: 0.3063 - val_output3_accuracy: 0.8381 - val_output3_dice_coef:
0.7762 - val_output3_loss: 0.3165 - learning_rate: 0.0010
Epoch 16/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.4280 - output1_acc
uracy: 0.8812 - output1_dice_coef: 0.8137 - output1_loss: 0.2419 - o
utput2_accuracy: 0.8792 - output2_dice_coef: 0.8107 - output2_loss:
0.2437 - output3_accuracy: 0.8707 - output3_dice_coef: 0.7964 - outp
ut3_loss: 0.2571 - val_loss: 0.4950 - val_output1_accuracy: 0.8590 -
val_output1_dice_coef: 0.7913 - val_output1_loss: 0.2832 - val_outpu
t2_accuracy: 0.8571 - val_output2_dice_coef: 0.7897 - val_output2_lo
ss: 0.2860 - val_output3_accuracy: 0.8475 - val_output3_dice_coef:
0.7727 - val_output3_loss: 0.3009 - learning_rate: 0.0010
Epoch 17/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.3748 - output1_acc
uracy: 0.8988 - output1_dice_coef: 0.8390 - output1_loss: 0.2103 - o
utput2_accuracy: 0.8967 - output2_dice_coef: 0.8352 - output2_loss:
0.2130 - output3_accuracy: 0.8870 - output3_dice_coef: 0.8154 - outp
ut3_loss: 0.2305 - val_loss: 0.4813 - val_output1_accuracy: 0.8621 -
val_output1_dice_coef: 0.8100 - val_output1_loss: 0.2770 - val_outpu
t2_accuracy: 0.8596 - val_output2_dice_coef: 0.8068 - val_output2_lo
ss: 0.2803 - val_output3_accuracy: 0.8483 - val_output3_dice_coef:
0.7881 - val_output3_loss: 0.2982 - learning_rate: 5.0000e-04
Epoch 18/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.3573 - output1_acc
uracy: 0.9023 - output1_dice_coef: 0.8478 - output1_loss: 0.2008 - o
utput2_accuracy: 0.9005 - output2_dice_coef: 0.8441 - output2_loss:
0.2033 - output3_accuracy: 0.8912 - output3_dice_coef: 0.8252 - outp
```

ut3_loss: 0.2208 - val_loss: 0.4746 - val_output1_accuracy: 0.8651 - val_output1_dice_coef: 0.8121 - val_output1_loss: 0.2728 - val_output2_accuracy: 0.8629 - val_output2_dice_coef: 0.8102 - val_output2_loss: 0.2773 - val_output3_accuracy: 0.8534 - val_output3_dice_coef: 0.7904 - val_output3_loss: 0.2927 - learning_rate: 5.0000e-04
Epoch 19/128
74/74 ━━━━━━━━━━━━━━━━━━━━ **3s** 43ms/step - loss: 0.3550 - output1_accuracy: 0.9041 - output1_dice_coef: 0.8485 - output1_loss: 0.1994 - output2_accuracy: 0.9020 - output2_dice_coef: 0.8448 - output2_loss: 0.2020 - output3_accuracy: 0.8918 - output3_dice_coef: 0.8258 - output3_loss: 0.2196 - val_loss: 0.4822 - val_output1_accuracy: 0.8676 - val_output1_dice_coef: 0.8037 - val_output1_loss: 0.2756 - val_output2_accuracy: 0.8654 - val_output2_dice_coef: 0.7984 - val_output2_loss: 0.2790 - val_output3_accuracy: 0.8545 - val_output3_dice_coef: 0.7810 - val_output3_loss: 0.2955 - learning_rate: 5.0000e-04
Epoch 20/128
74/74 ━━━━━━━━━━━━━━━━━━━━ **3s** 42ms/step - loss: 0.3402 - output1_accuracy: 0.9083 - output1_dice_coef: 0.8546 - output1_loss: 0.1909 - output2_accuracy: 0.9065 - output2_dice_coef: 0.8515 - output2_loss: 0.1931 - output3_accuracy: 0.8972 - output3_dice_coef: 0.8342 - output3_loss: 0.2106 - val_loss: 0.4866 - val_output1_accuracy: 0.8661 - val_output1_dice_coef: 0.7933 - val_output1_loss: 0.2782 - val_output2_accuracy: 0.8643 - val_output2_dice_coef: 0.7895 - val_output2_loss: 0.2803 - val_output3_accuracy: 0.8544 - val_output3_dice_coef: 0.7770 - val_output3_loss: 0.2953 - learning_rate: 5.0000e-04
Epoch 21/128
74/74 ━━━━━━━━━━━━━━━━━━━━ **3s** 45ms/step - loss: 0.3253 - output1_accuracy: 0.9127 - output1_dice_coef: 0.8621 - output1_loss: 0.1825 - output2_accuracy: 0.9110 - output2_dice_coef: 0.8585 - output2_loss: 0.1845 - output3_accuracy: 0.9008 - output3_dice_coef: 0.8384 - output3_loss: 0.2031 - val_loss: 0.5348 - val_output1_accuracy: 0.8522 - val_output1_dice_coef: 0.8142 - val_output1_loss: 0.3135 - val_output2_accuracy: 0.8514 - val_output2_dice_coef: 0.8119 - val_output2_loss: 0.3090 - val_output3_accuracy: 0.8468 - val_output3_dice_coef: 0.7975 - val_output3_loss: 0.3143 - learning_rate: 5.0000e-04
Epoch 22/128
74/74 ━━━━━━━━━━━━━━━━━━━━ **3s** 43ms/step - loss: 0.3084 - output1_accuracy: 0.9184 - output1_dice_coef: 0.8687 - output1_loss: 0.1720 - output2_accuracy: 0.9165 - output2_dice_coef: 0.8649 - output2_loss: 0.1750 - output3_accuracy: 0.9051 - output3_dice_coef: 0.8435 - output3_loss: 0.1960 - val_loss: 0.4724 - val_output1_accuracy: 0.8719 - val_output1_dice_coef: 0.8274 - val_output1_loss: 0.2727 - val_output2_accuracy: 0.8696 - val_output2_dice_coef: 0.8243 - val_output2_loss: 0.2747 - val_output3_accuracy: 0.8588 - val_output3_dice_coef: 0.8040 - val_output3_loss: 0.2875 - learning_rate: 2.5000e-04
Epoch 23/128
74/74 ━━━━━━━━━━━━━━━━━━━━ **3s** 42ms/step - loss: 0.2816 - output1_accuracy: 0.9253 - output1_dice_coef: 0.8814 - output1_loss: 0.1566 - output2_accuracy: 0.9236 - output2_dice_coef: 0.8774 - output2_loss: 0.1594 - output3_accuracy: 0.9126 - output3_dice_coef: 0.8568 - output3_loss: 0.1802 - val_loss: 0.4758 - val_output1_accuracy: 0.8732 - val_output1_dice_coef: 0.8273 - val_output1_loss: 0.2751 - val_output2_accuracy: 0.8716 - val_output2_dice_coef: 0.8232 - val_output2_loss: 0.2764 - val_output3_accuracy: 0.8597 - val_output3_dice_coef: 0.8024 - val_output3_loss: 0.2898 - learning_rate: 2.5000e-04
Epoch 24/128

```
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.2734 - output1_acc
uracy: 0.9280 - output1_dice_coef: 0.8859 - output1_loss: 0.1521 - o
utput2_accuracy: 0.9261 - output2_dice_coef: 0.8819 - output2_loss:
0.1550 - output3_accuracy: 0.9145 - output3_dice_coef: 0.8598 - outp
ut3_loss: 0.1769 - val_loss: 0.5296 - val_output1_accuracy: 0.8619 -
val_output1_dice_coef: 0.8255 - val_output1_loss: 0.3088 - val_outpu
t2_accuracy: 0.8598 - val_output2_dice_coef: 0.8222 - val_output2_lo
ss: 0.3080 - val_output3_accuracy: 0.8513 - val_output3_dice_coef:
0.8039 - val_output3_loss: 0.3110 - learning_rate: 2.5000e-04
Epoch 25/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - loss: 0.2620 - output1_acc
uracy: 0.9310 - output1_dice_coef: 0.8908 - output1_loss: 0.1453 - o
utput2_accuracy: 0.9290 - output2_dice_coef: 0.8868 - output2_loss:
0.1485 - output3_accuracy: 0.9170 - output3_dice_coef: 0.8648 - outp
ut3_loss: 0.1707 - val_loss: 0.4931 - val_output1_accuracy: 0.8682 -
val_output1_dice_coef: 0.8243 - val_output1_loss: 0.2845 - val_outpu
t2_accuracy: 0.8661 - val_output2_dice_coef: 0.8217 - val_output2_lo
ss: 0.2869 - val_output3_accuracy: 0.8560 - val_output3_dice_coef:
0.8028 - val_output3_loss: 0.2990 - learning_rate: 1.2500e-04
Epoch 26/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.2563 - output1_acc
uracy: 0.9326 - output1_dice_coef: 0.8918 - output1_loss: 0.1420 - o
utput2_accuracy: 0.9308 - output2_dice_coef: 0.8877 - output2_loss:
0.1452 - output3_accuracy: 0.9191 - output3_dice_coef: 0.8658 - outp
ut3_loss: 0.1674 - val_loss: 0.4923 - val_output1_accuracy: 0.8727 -
val_output1_dice_coef: 0.8305 - val_output1_loss: 0.2862 - val_outpu
t2_accuracy: 0.8707 - val_output2_dice_coef: 0.8268 - val_output2_lo
ss: 0.2868 - val_output3_accuracy: 0.8606 - val_output3_dice_coef:
0.8069 - val_output3_loss: 0.2943 - learning_rate: 1.2500e-04
Epoch 27/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.2522 - output1_acc
uracy: 0.9341 - output1_dice_coef: 0.8944 - output1_loss: 0.1396 - o
utput2_accuracy: 0.9323 - output2_dice_coef: 0.8903 - output2_loss:
0.1429 - output3_accuracy: 0.9200 - output3_dice_coef: 0.8683 - outp
ut3_loss: 0.1653 - val_loss: 0.5126 - val_output1_accuracy: 0.8691 -
val_output1_dice_coef: 0.8307 - val_output1_loss: 0.2994 - val_outpu
t2_accuracy: 0.8668 - val_output2_dice_coef: 0.8269 - val_output2_lo
ss: 0.2979 - val_output3_accuracy: 0.8572 - val_output3_dice_coef:
0.8067 - val_output3_loss: 0.3009 - learning_rate: 1.2500e-04
Epoch 28/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.2477 - output1_acc
uracy: 0.9350 - output1_dice_coef: 0.8969 - output1_loss: 0.1366 - o
utput2_accuracy: 0.9334 - output2_dice_coef: 0.8930 - output2_loss:
0.1399 - output3_accuracy: 0.9207 - output3_dice_coef: 0.8699 - outp
ut3_loss: 0.1637 - val_loss: 0.5050 - val_output1_accuracy: 0.8703 -
val_output1_dice_coef: 0.8302 - val_output1_loss: 0.2942 - val_outpu
t2_accuracy: 0.8687 - val_output2_dice_coef: 0.8264 - val_output2_lo
ss: 0.2936 - val_output3_accuracy: 0.8579 - val_output3_dice_coef:
0.8066 - val_output3_loss: 0.2993 - learning_rate: 6.2500e-05
Epoch 29/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.2426 - output1_acc
uracy: 0.9368 - output1_dice_coef: 0.9001 - output1_loss: 0.1339 - o
utput2_accuracy: 0.9349 - output2_dice_coef: 0.8957 - output2_loss:
0.1372 - output3_accuracy: 0.9227 - output3_dice_coef: 0.8727 - outp
ut3_loss: 0.1603 - val_loss: 0.4929 - val_output1_accuracy: 0.8721 -
val_output1_dice_coef: 0.8288 - val_output1_loss: 0.2853 - val_outpu
```

```
t2_accuracy: 0.8701 - val_output2_dice_coef: 0.8254 - val_output2_lo
ss: 0.2876 - val_output3_accuracy: 0.8589 - val_output3_dice_coef:
0.8058 - val_output3_loss: 0.2966 - learning_rate: 6.2500e-05
Epoch 30/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 43ms/step - loss: 0.2406 - output1_acc
uracy: 0.9370 - output1_dice_coef: 0.8993 - output1_loss: 0.1326 - o
utput2_accuracy: 0.9352 - output2_dice_coef: 0.8951 - output2_loss:
0.1361 - output3_accuracy: 0.9233 - output3_dice_coef: 0.8723 - outp
ut3_loss: 0.1594 - val_loss: 0.5100 - val_output1_accuracy: 0.8685 -
val_output1_dice_coef: 0.8289 - val_output1_loss: 0.2972 - val_outpu
t2_accuracy: 0.8670 - val_output2_dice_coef: 0.8254 - val_output2_lo
ss: 0.2965 - val_output3_accuracy: 0.8577 - val_output3_dice_coef:
0.8059 - val_output3_loss: 0.3007 - learning_rate: 6.2500e-05
Epoch 31/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 4s 54ms/step - loss: 0.2397 - output1_acc
uracy: 0.9377 - output1_dice_coef: 0.9000 - output1_loss: 0.1319 - o
utput2_accuracy: 0.9357 - output2_dice_coef: 0.8961 - output2_loss:
0.1353 - output3_accuracy: 0.9235 - output3_dice_coef: 0.8731 - outp
ut3_loss: 0.1593 - val_loss: 0.5051 - val_output1_accuracy: 0.8705 -
val_output1_dice_coef: 0.8300 - val_output1_loss: 0.2938 - val_outpu
t2_accuracy: 0.8688 - val_output2_dice_coef: 0.8263 - val_output2_lo
ss: 0.2944 - val_output3_accuracy: 0.8585 - val_output3_dice_coef:
0.8068 - val_output3_loss: 0.2999 - learning_rate: 3.1250e-05
Epoch 32/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 4s 53ms/step - loss: 0.2391 - output1_acc
uracy: 0.9378 - output1_dice_coef: 0.8999 - output1_loss: 0.1320 - o
utput2_accuracy: 0.9361 - output2_dice_coef: 0.8955 - output2_loss:
0.1354 - output3_accuracy: 0.9238 - output3_dice_coef: 0.8723 - outp
ut3_loss: 0.1592 - val_loss: 0.5062 - val_output1_accuracy: 0.8709 -
val_output1_dice_coef: 0.8304 - val_output1_loss: 0.2946 - val_outpu
t2_accuracy: 0.8693 - val_output2_dice_coef: 0.8269 - val_output2_lo
ss: 0.2948 - val_output3_accuracy: 0.8582 - val_output3_dice_coef:
0.8072 - val_output3_loss: 0.2999 - learning_rate: 3.1250e-05
Epoch 33/128
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.2349 - output1_acc
uracy: 0.9389 - output1_dice_coef: 0.9030 - output1_loss: 0.1293 - o
utput2_accuracy: 0.9370 - output2_dice_coef: 0.8987 - output2_loss:
0.1329 - output3_accuracy: 0.9245 - output3_dice_coef: 0.8759 - outp
ut3_loss: 0.1566 - val_loss: 0.5041 - val_output1_accuracy: 0.8707 -
val_output1_dice_coef: 0.8296 - val_output1_loss: 0.2928 - val_outpu
t2_accuracy: 0.8690 - val_output2_dice_coef: 0.8261 - val_output2_lo
ss: 0.2937 - val_output3_accuracy: 0.8580 - val_output3_dice_coef:
0.8067 - val_output3_loss: 0.2999 - learning_rate: 3.1250e-05
Epoch 33: early stopping
Restoring model weights from the end of the best epoch: 18.
```






```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
```

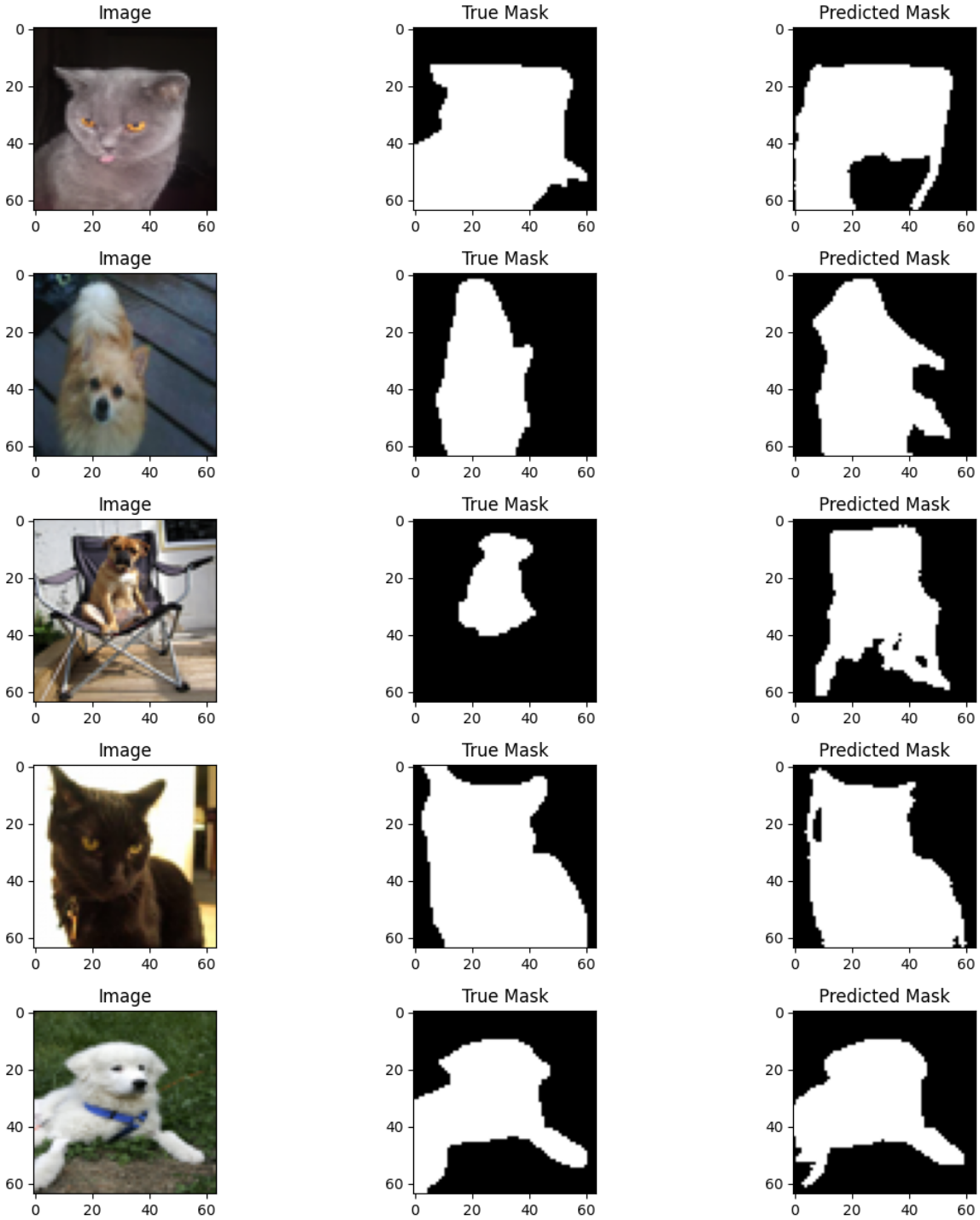
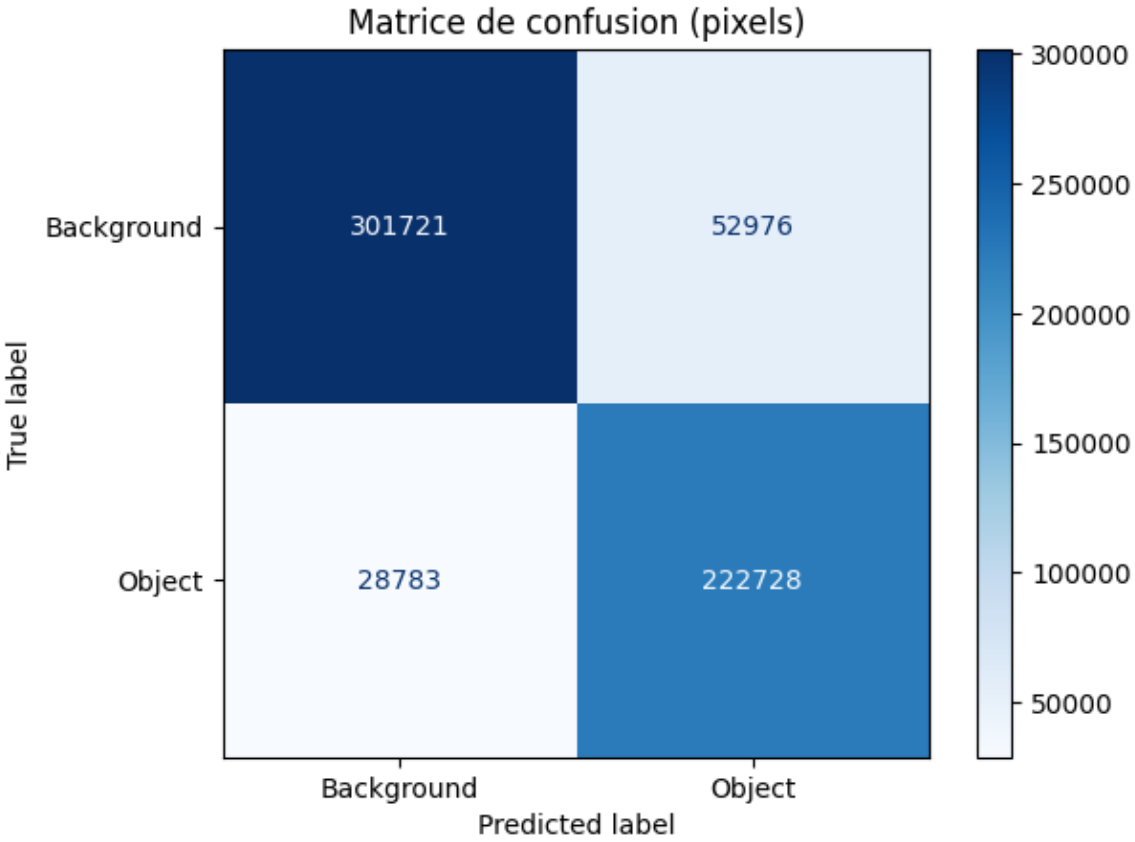


Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask
Image
True Mask
Predicted Mask




```
Classification report (par pixel):
              precision    recall  f1-score   support

  Background       0.91      0.85      0.88    354697
      Object       0.81      0.89      0.84    251511

    accuracy                           0.87    606208
   macro avg       0.86      0.87      0.86    606208
weighted avg       0.87      0.87      0.87    606208


Précision globale (pixel accuracy): 0.8651
Temps d'entraînement : 108.47 secondes
```