Participez à la conception d’une voiture autonome

Future Vision Transport

lf-Driving
48
mph
Autonomous
Mode
NETWORK
DRIVER
INFORMATION

Sommaire

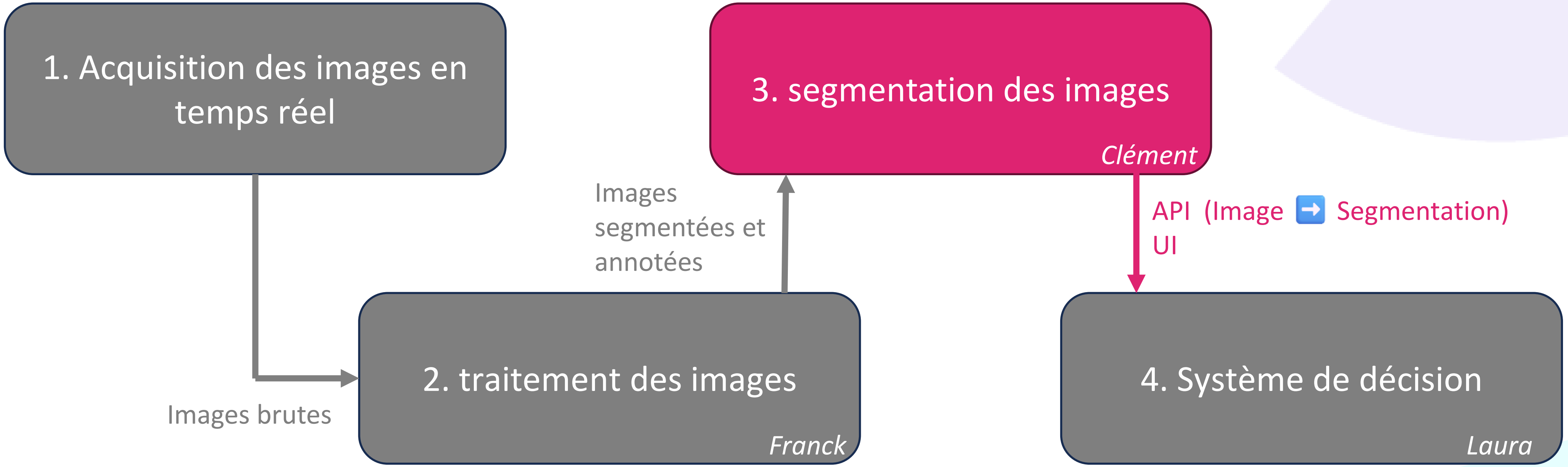
Future Vision Transport

PARTIE 1 – INTRODUCTION

Future Vision Transport

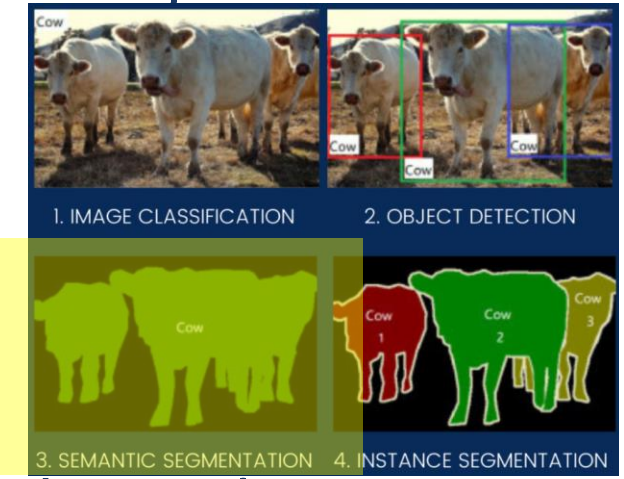
Le projet de l’entreprise

- Conception systèmes embarqués de computer vision
- Cible : véhicules autonomes
- Équipe projet divisée en plusieurs parties :

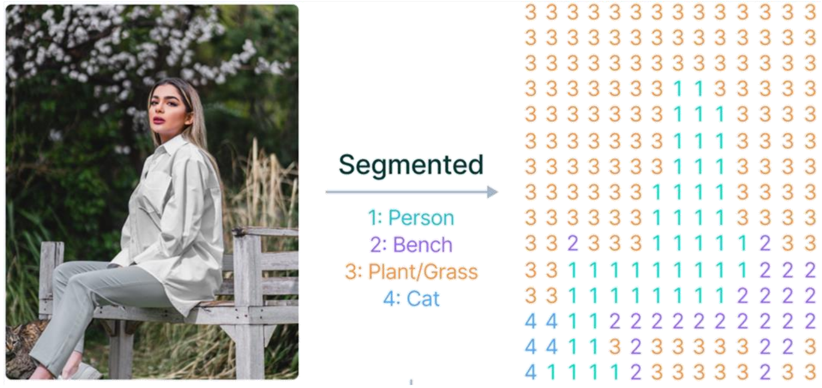


Qu’est-ce que la segmentation ?

- Intégrée au champ de la *Computer Vision* :



- Classification à l’échelle du pixel :



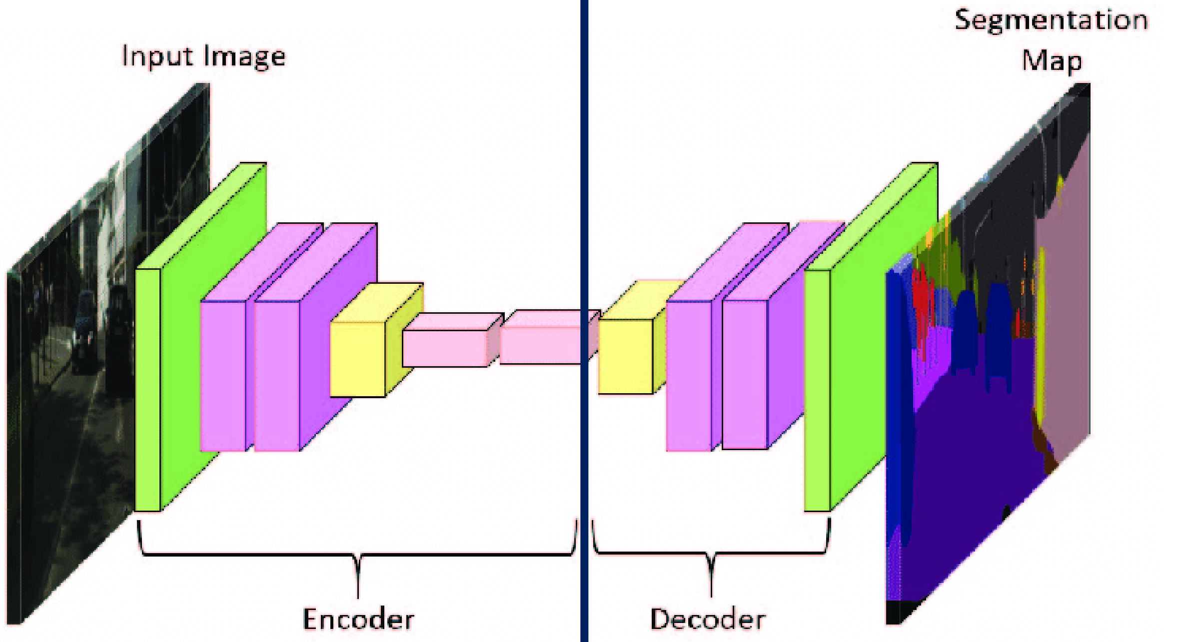
Fonctionnement général

Architecture : *Encoder* *Decoder*

Extraire les features importantes

Basé sur des modèles connus sur la classification d'image (sans les couches de décision)
➡️ *Backbone*

Downsampling



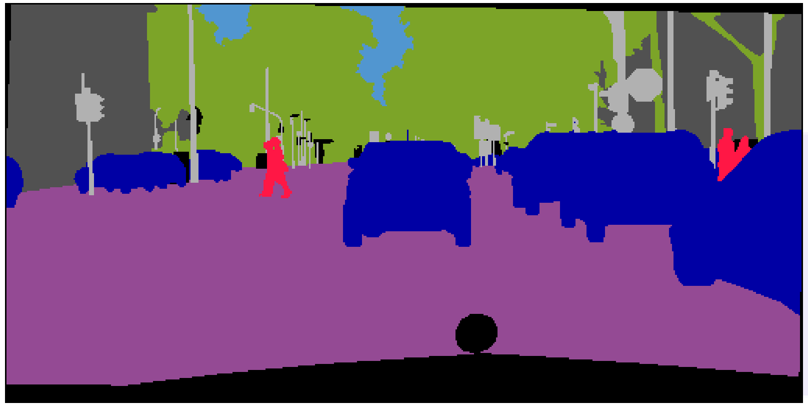
Upsampling

Revenir aux dimensions initiales

Autant de canaux que de classes à détecter

Les données à notre disposition

- Images : `P8_Cityscapes_leftImg8bit_trainvaltest`
- Masks : `P8_Cityscapes_gtFine_trainvaltest`



- Prédécoupage :
  - Entraînement : 2975 images / masks
  - Validation : 500 images / masks
  - Test : 1525 images / ~~masks~~

Réduire le nombre de catégories

name	id	trainId	category	catId
'unlabeled'	0	255	'void'	0
'ego vehicle'	1	255	'void'	0
'rectification border'	2	255	'void'	0
'out of roi'	3	255	'void'	0
'static'	4	255	'void'	0
'dynamic'	5	255	'void'	0
'ground'	6	255	'void'	0
'road'	7	0	'flat'	1
'sidewalk'	8	1	'flat'	1
'parking'	9	255	'flat'	1
'rail track'	10	255	'flat'	1
'building'	11	2	'construction'	2
'wall'	12	3	'construction'	2
'fence'	13	4	'construction'	2
'guard rail'	14	255	'construction'	2
'bridge'	15	255	'construction'	2
'tunnel'	16	255	'construction'	2
'pole'	17	5	'object'	3

name	id	trainId	category	catId
'polegroup'	18	255	'object'	3
'traffic light'	19	6	'object'	3
'traffic sign'	20	7	'object'	3
'vegetation'	21	8	'nature'	4
'terrain'	22	9	'nature'	4
'sky'	23	10	'sky'	5
'person'	24	11	'human'	6
'rider'	25	12	'human'	6
'car'	26	13	'vehicle'	7
'truck'	27	14	'vehicle'	7
'bus'	28	15	'vehicle'	7
'caravan'	29	255	'vehicle'	7
'trailer'	30	255	'vehicle'	7
'train'	31	16	'vehicle'	7
'motorcycle'	32	17	'vehicle'	7
'bicycle'	33	18	'vehicle'	7
'license plate'	-1	-1	'vehicle'	7

Les outils utilisés

- Mise à disposition des images Keras
- Export du modèle TensorFlow
- Modélisation Segmentation Models
- Augmentation de données Albumentations
- Création de l’API FastAPI
- Création de l’UI Flask
- Conteneurs exécutables docker
- Déploiement
- Intégration continue GitHub
- Déploiement continue GitHub Actions +

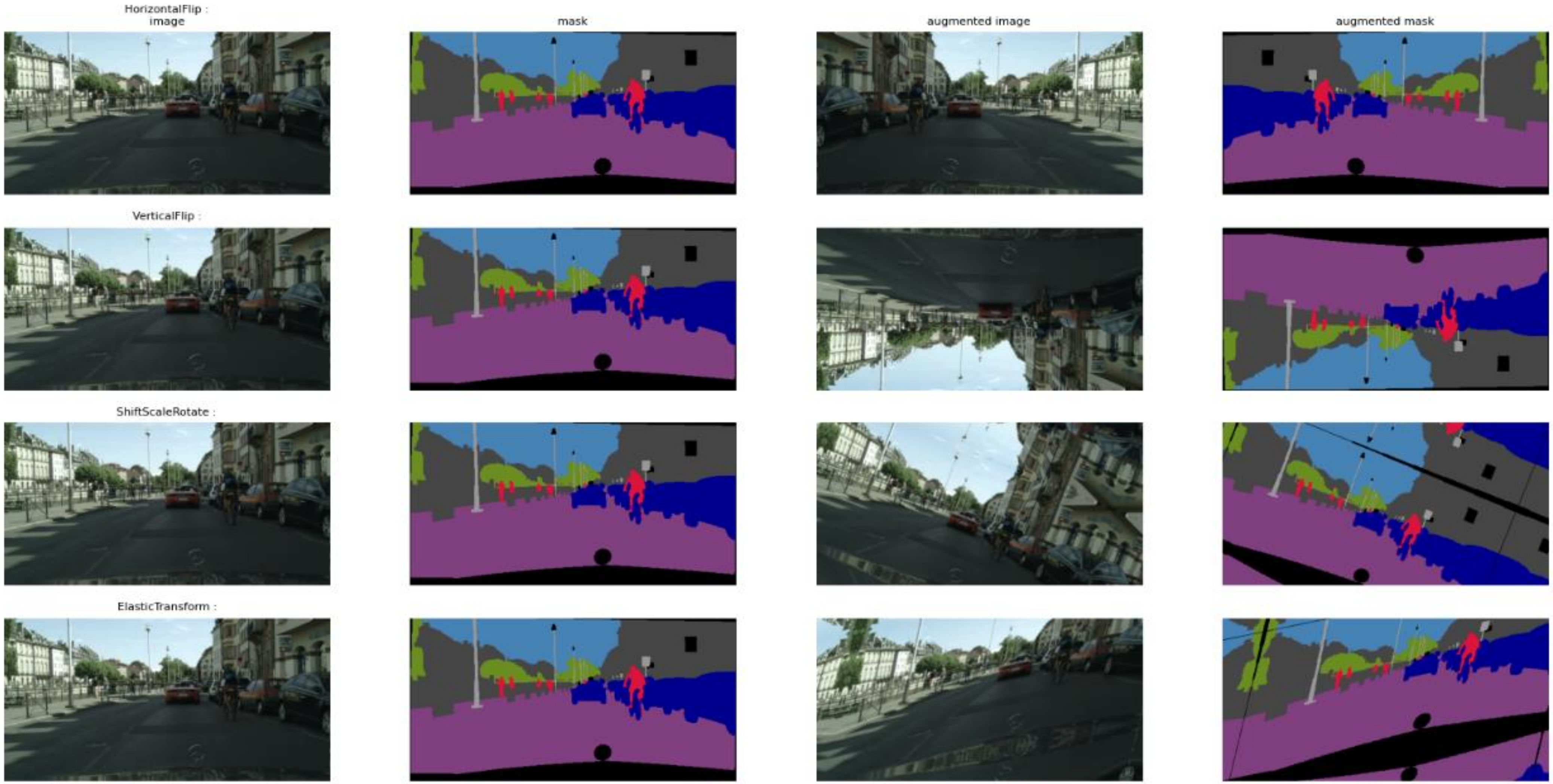
Future Vision Transport

PARTIE 2 – PREPROCESSING

Générateur de données

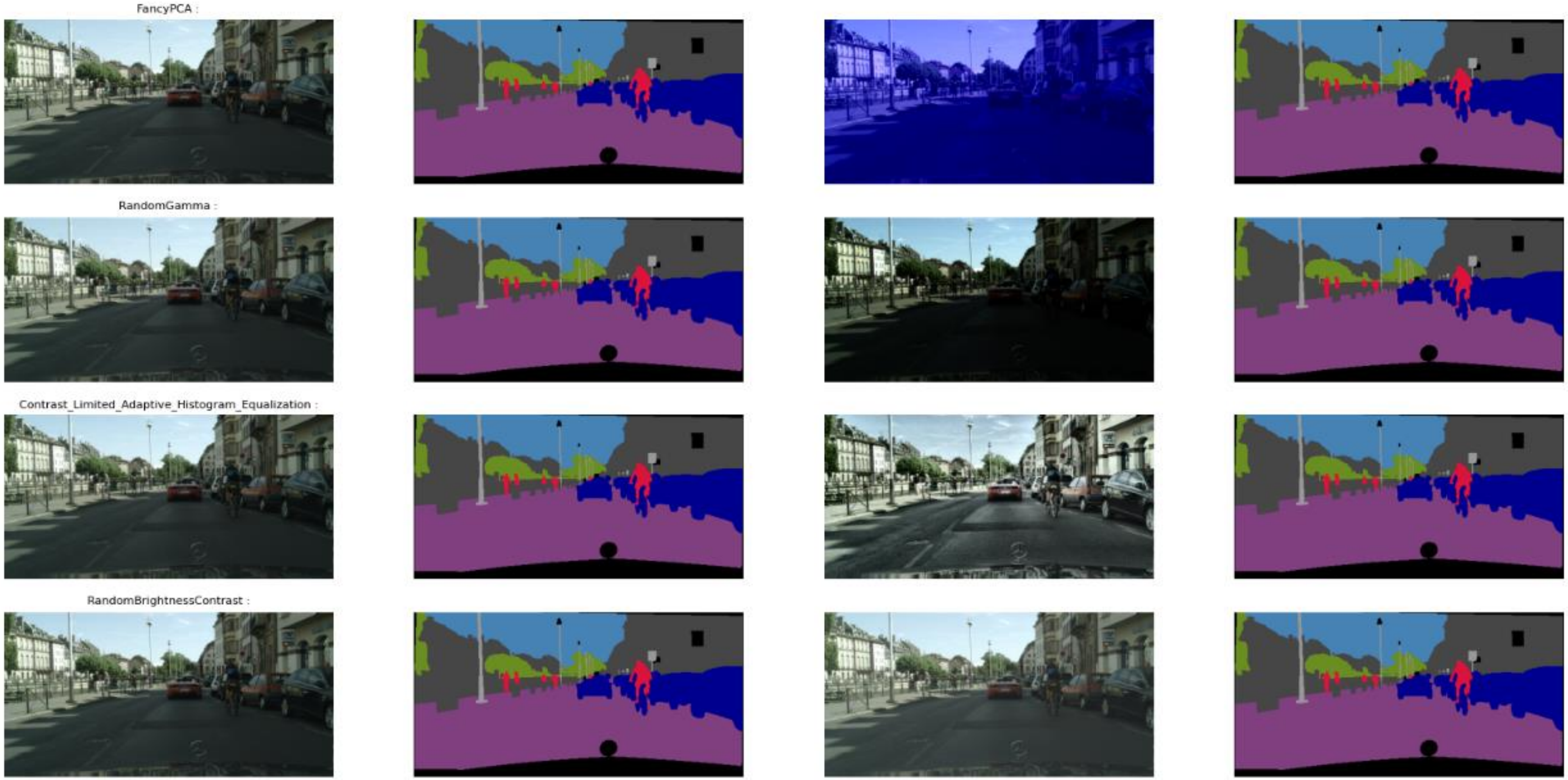
- Dataset très volumineux (11 Go)
- Sous-classe de `keras.utils.Sequence`
  - 🔽 chemins d’accès des images et des masks
  - (facultatif) échantillonnage `n_images`
  - (facultatif) `train_test_split`
  - Lecture images et les masks par lot :
    - à la taille souhaitée `batch_size`
    - aux dimensions souhaitées `image_size`
  - (facultatif) augmentation des données
  - (facultatif) prétraitement images
  - (facultatif) mappage des classes en macro-catégories
  - labels du mask dans leurs propres canaux (256 x 512) ➡️ (256 x 512 x 8)
  - mise à disposition du batch créé
  - (facultatif) après chaque `epoch` ➡️ mélange les chemins d’accès des images/masks

Augmentation des données - exploration

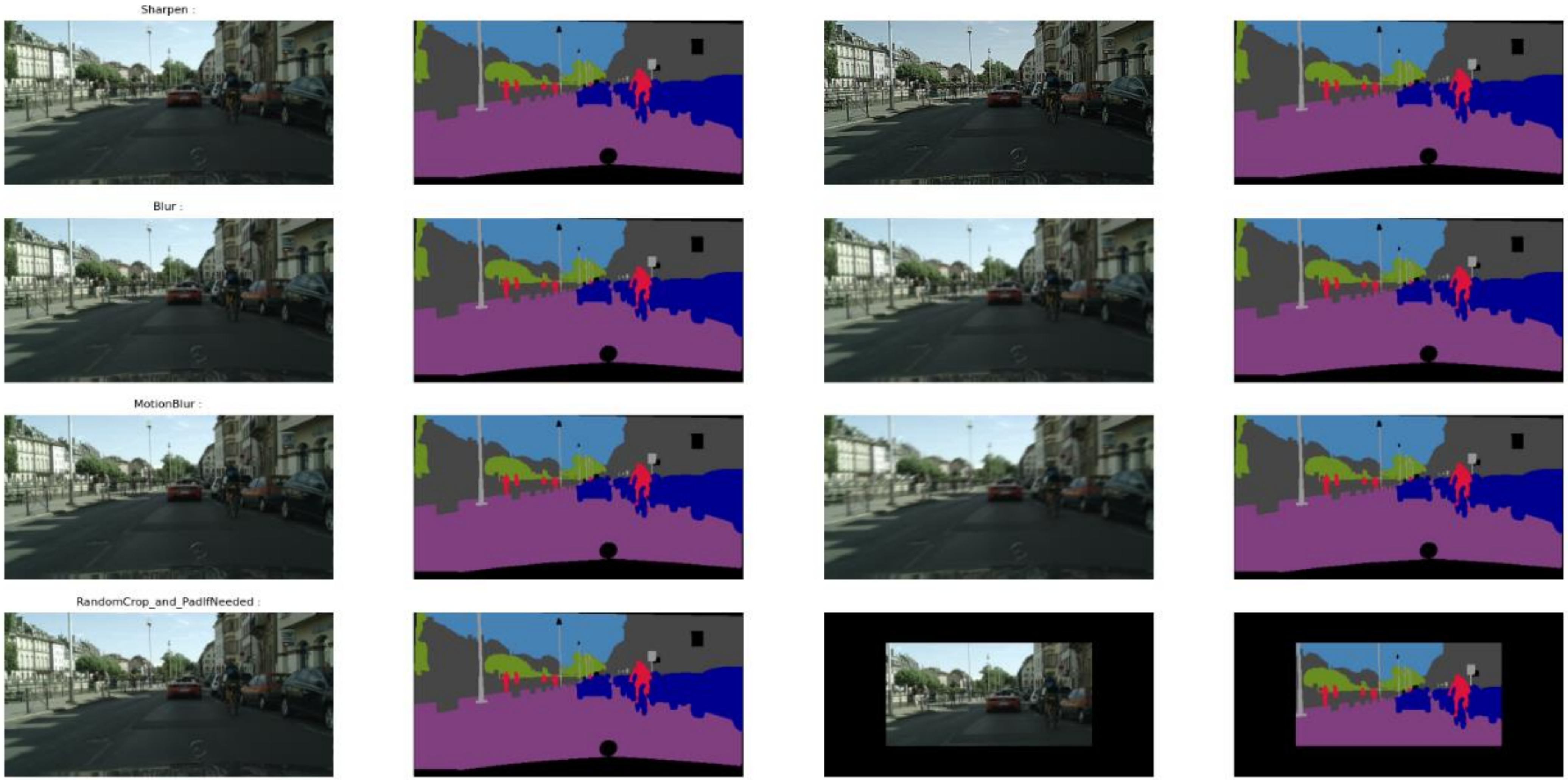


Future Vision Transport

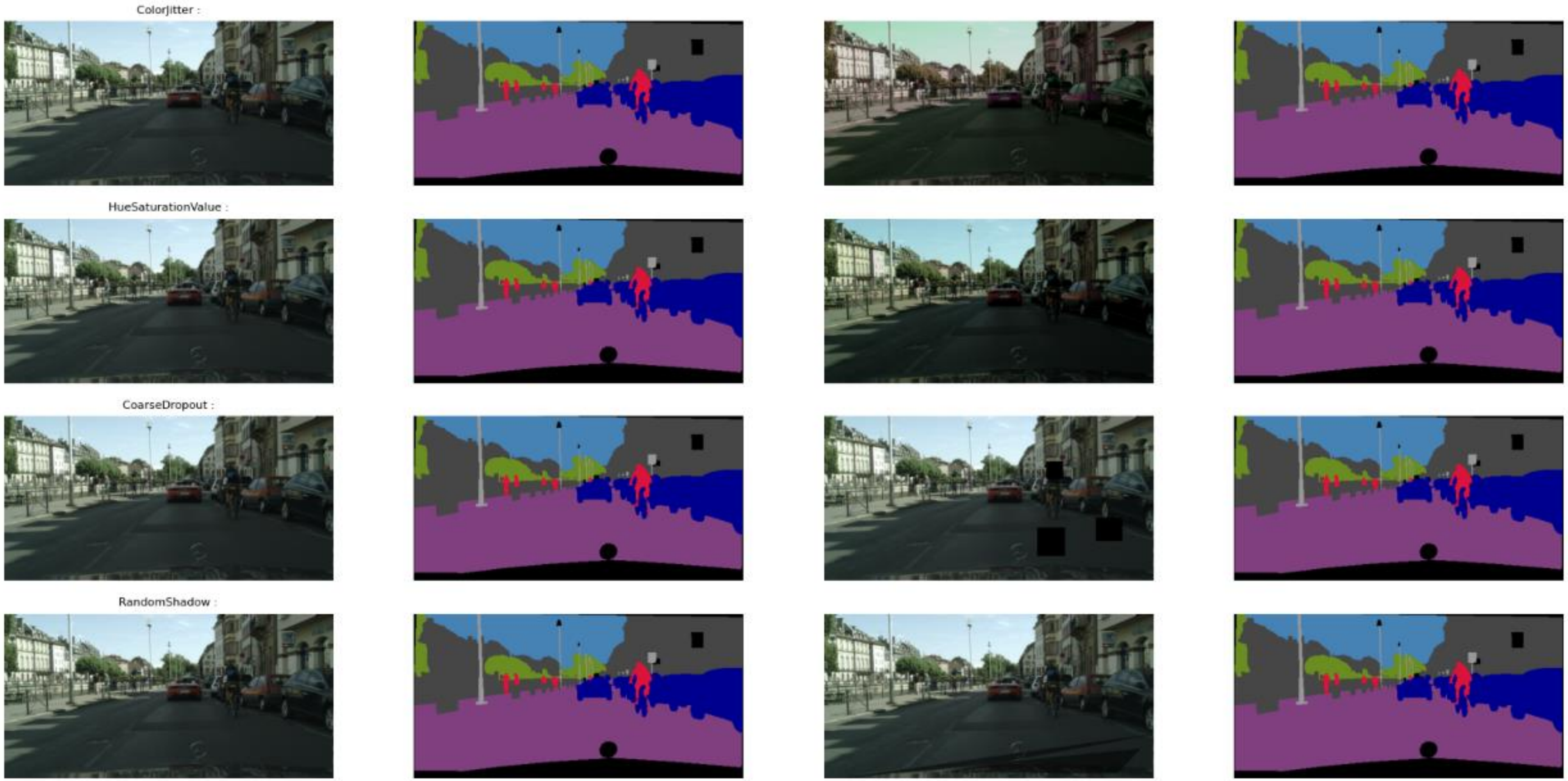
Augmentation des données - exploration



Augmentation des données - exploration



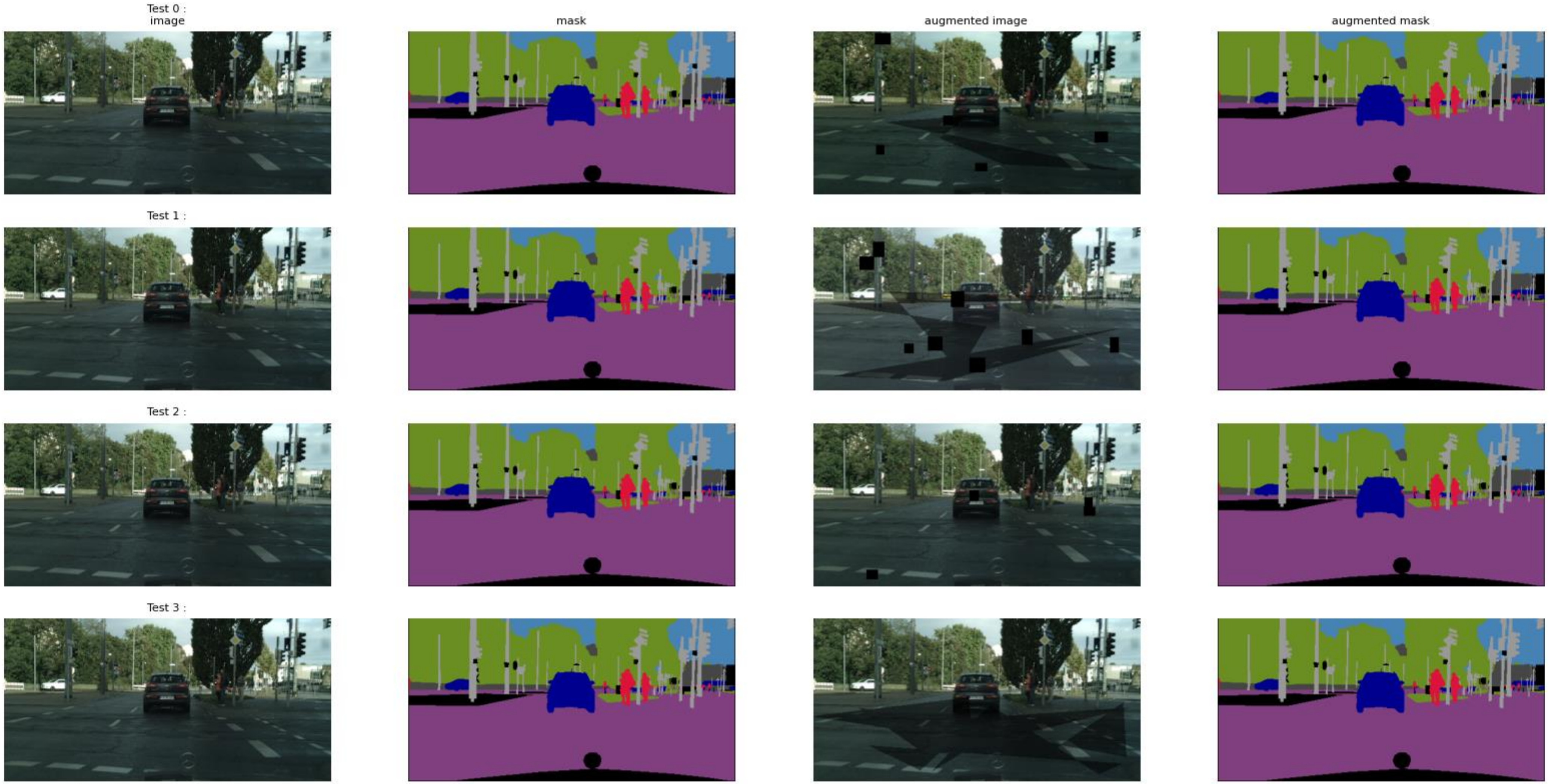
Augmentation des données - exploration



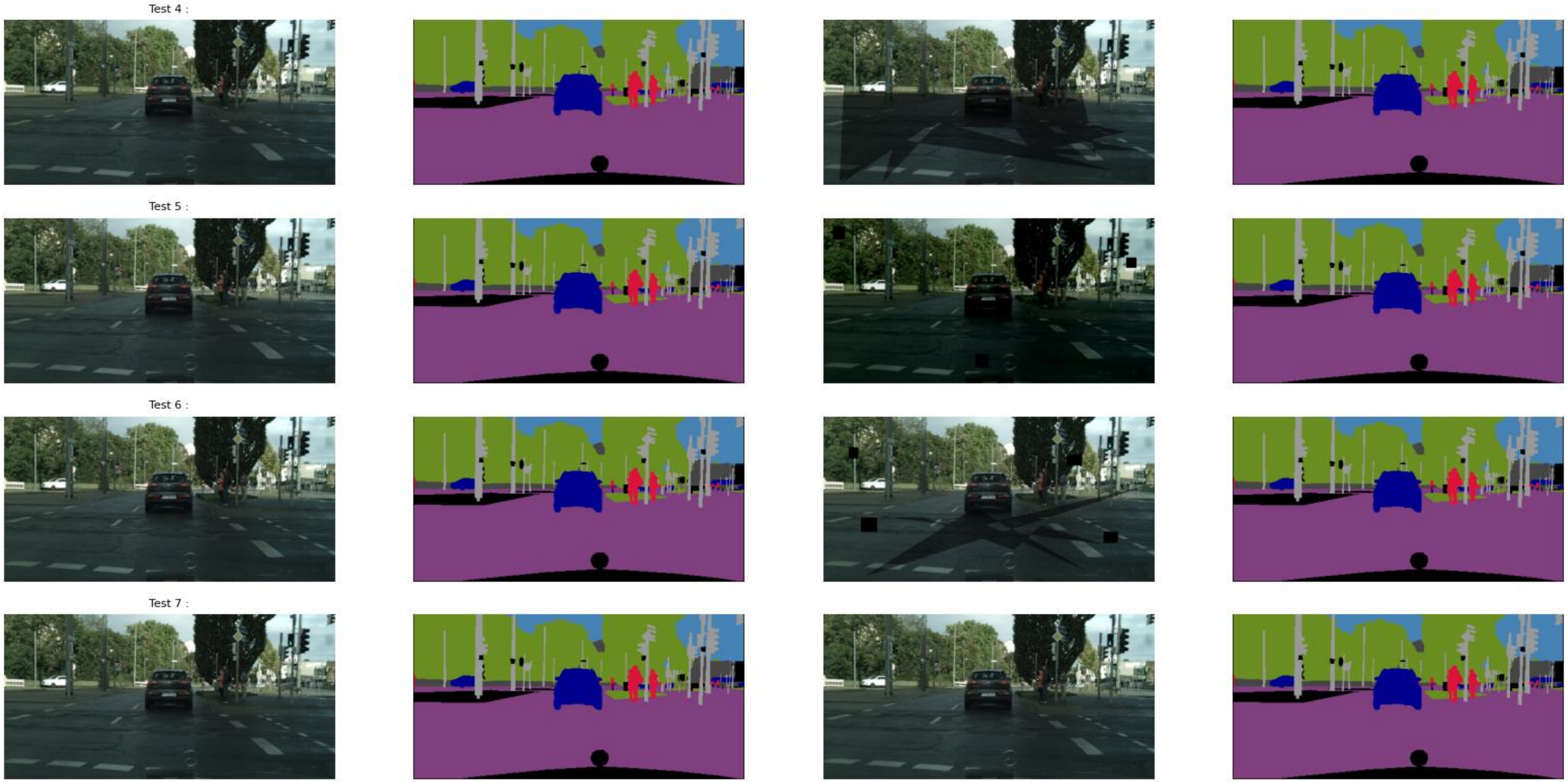
Augmentation des données – solution retenue

```
# define augmentations for training
list_of_transforms = [
    A.OneOf(
        [
            A.FancyPCA(p=1, alpha=1),
            A.HueSaturationValue(p=1, hue_shift_limit=20, sat_shift_limit=30, val_shift_limit=20),
            A.ColorJitter(p=1, brightness=0.2, contrast=0.2, saturation=0.2),
        ],
        p=0.5
    ),
    A.RandomShadow(
        p=0.5,
        shadow_roi=(
            0, 0.4,
            1, 1
            ),
        num_shadows_lower=1,
        num_shadows_upper=4,
        shadow_dimension=5
        ),
    A.CoarseDropout(
        p=0.5,
        min_holes=2,
        max_holes=8,
        min_height=0.05,
        max_height=0.1,
        min_width=0.025,
        max_width=0.05
        )
]
```

Augmentation des données – solution retenue



Augmentation des données - solution retenue



PARTIE 3 – Modélisation

Métrique d’évaluation

- Mean IoU score (indice de Jaccard) :

$$IoU = \frac{TP}{(TP + FP + FN)}$$

Ground Truth Mask   Predicted Mask

- Dice score :

$$Dice = \frac{2TP}{2TP + FP + FN}$$

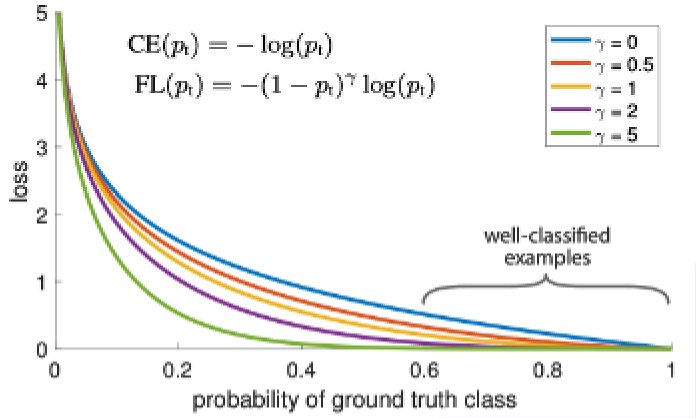
- Training time

Fonction de perte

- Focal Loss

$$\mathrm{CE}\left(p,\hat{p}\right) = \cancel{-\left(p\log\left(\hat{p}\right) + (1-p)\right.}\log\left(1-\hat{p}\right)\Big)$$

$$\mathrm{FL}\left(p,\hat{p}\right) = -\left(\alpha(1-\hat{p})^{\gamma}p\log\left(\hat{p}\right) + (1-\alpha)\hat{p}^{\gamma}(1-p)\log\left(1-\hat{p}\right)\right)$$



+

- Dice Loss

$$\mathrm{DL}\left(p,\hat{p}\right) = 1 - \frac{2\sum p_{h,w}\hat{p}_{h,w}}{\sum p_{h,w} + \sum \hat{p}_{h,w}}$$

Avec ou sans l’argument `class_weights`

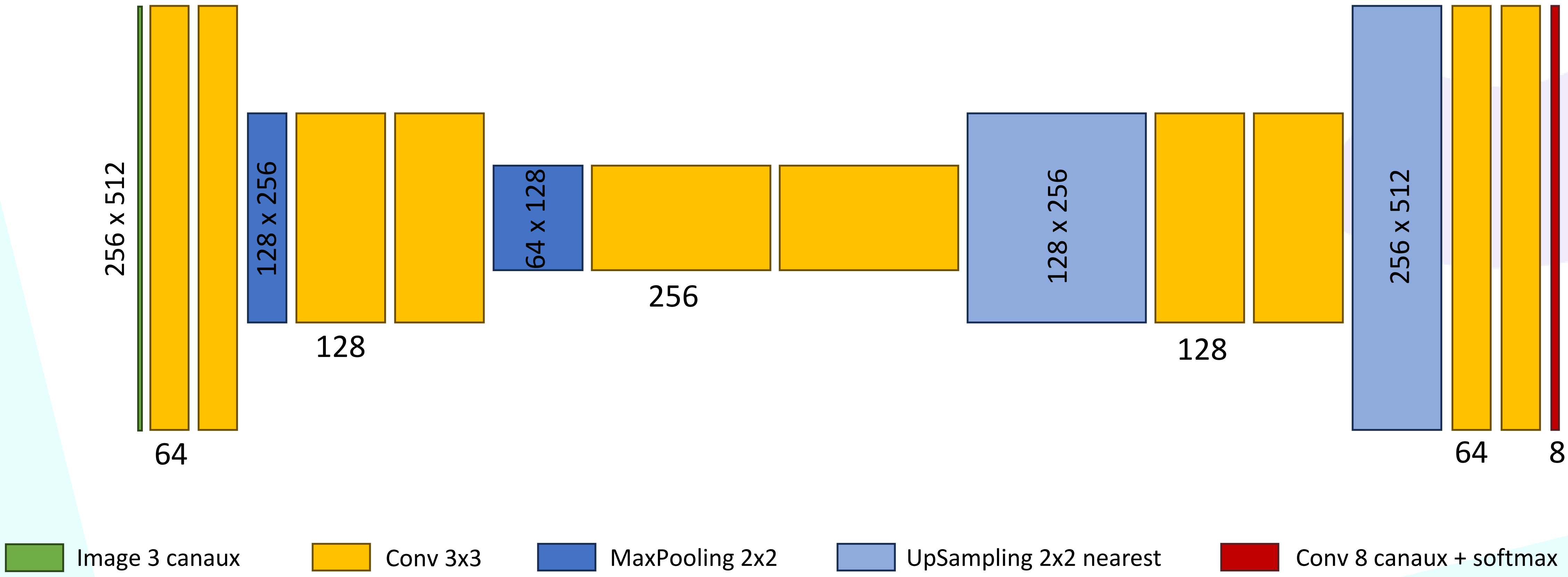
Choix pour la modélisation

- Réduire le temps d'apprentissage :
  - Limitation du jeu d'Entraînement : ~~2975~~ ➡️ 512 images
  - Limitation de la taille des images : (512 x 1024) ➡️ (256 x 512)

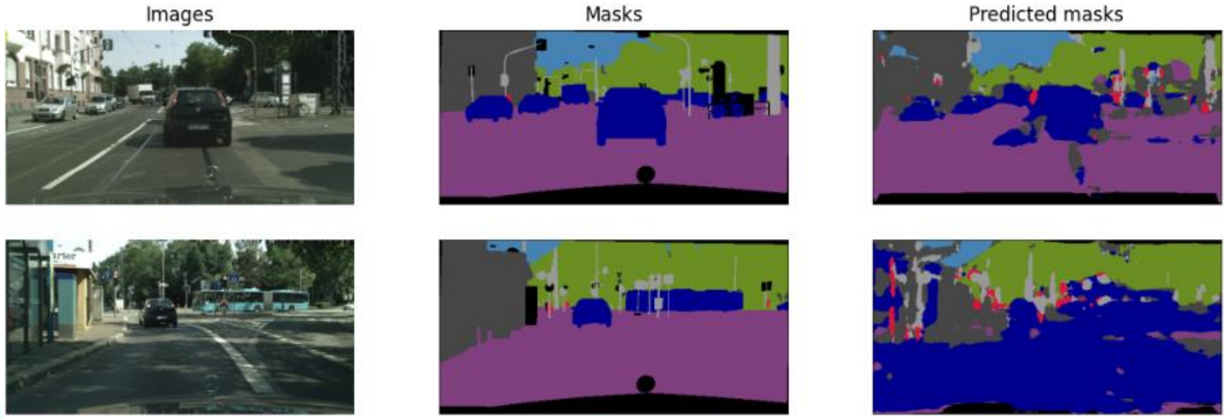
- Le nombre d'*epochs*
  - Callback `EarlyStopping`
  - Patience = 3 epochs
  - Warm-up = 10 epochs

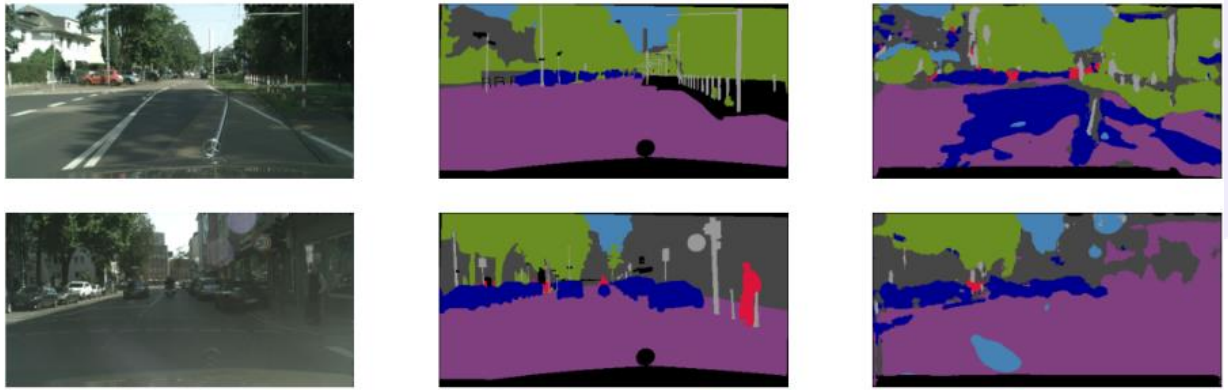
Baseline

- Modèle de segmentation sémantique simple :



Baseline

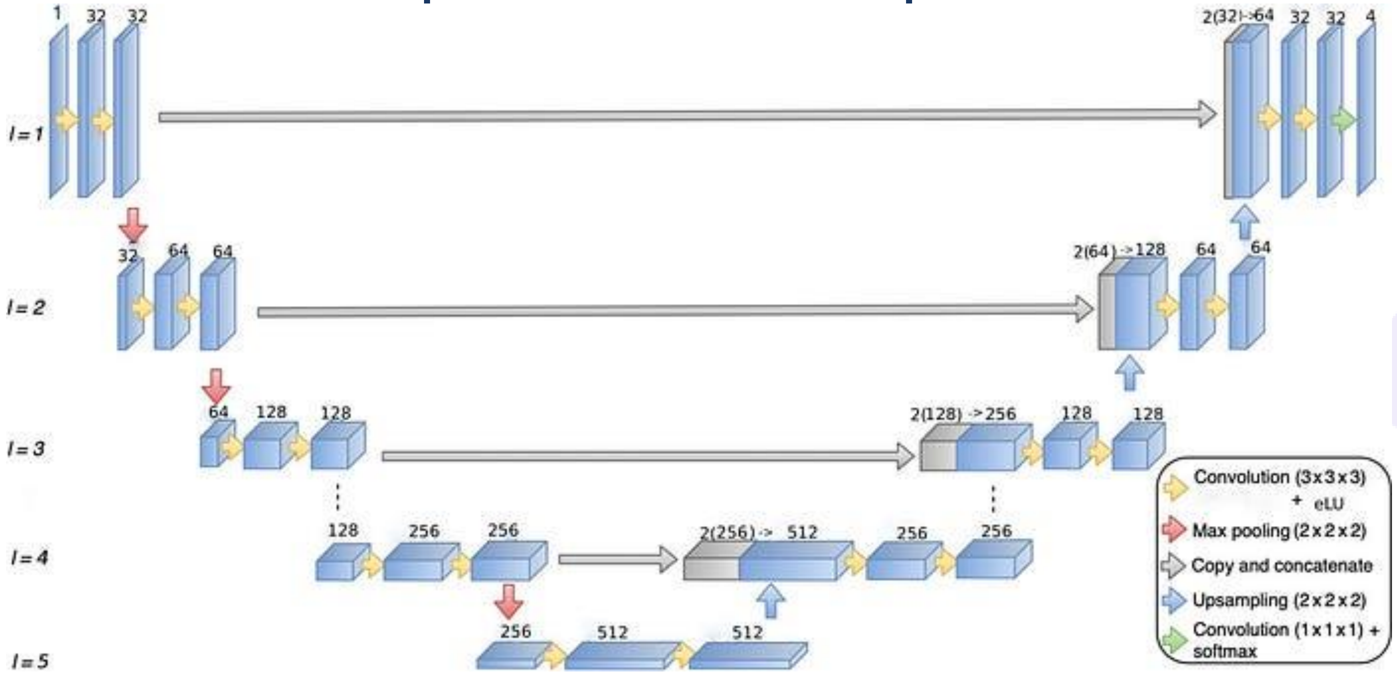




model	backbone	loss_function	val_iou	val_f_score	training_time	augmentation
baseline		dice+focal	0.388	0.490	259.22 min	No

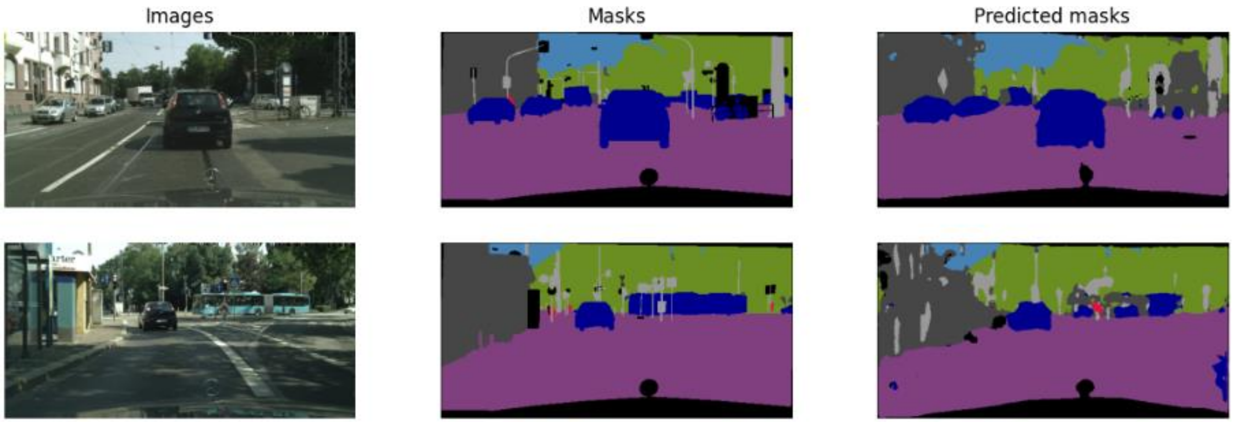
Architecture U-net

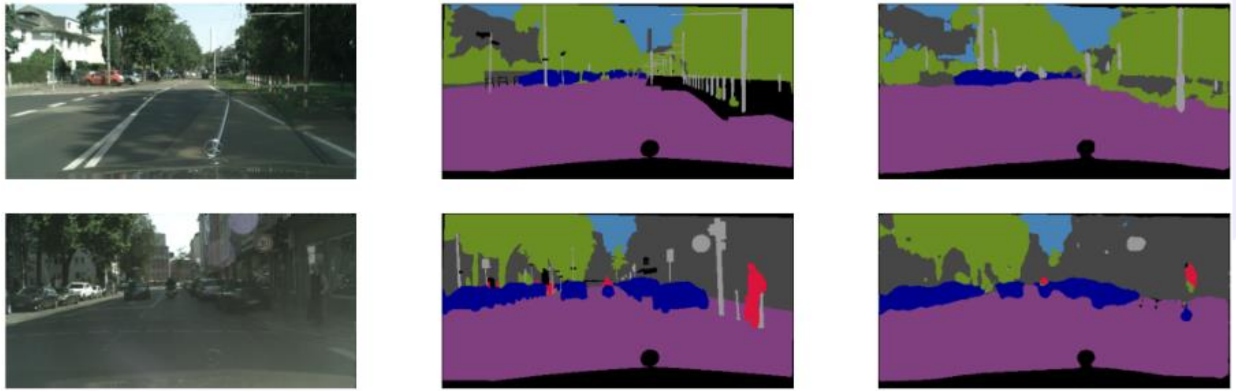
- *Skip connections* : mieux prendre en compte l'information spatiale



- *Backbone* : `VGG16` et `Resnet34`

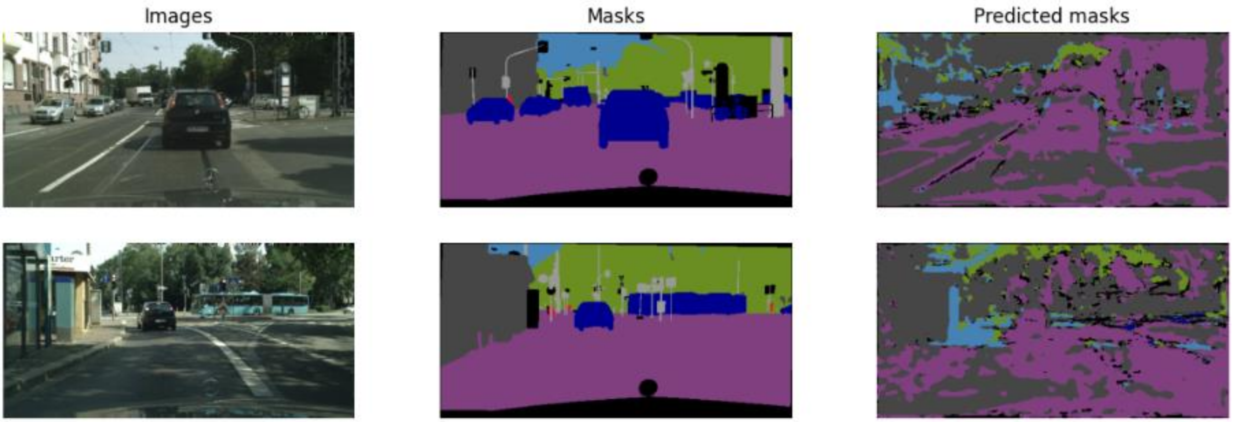
Architecture U-net – Resnet34

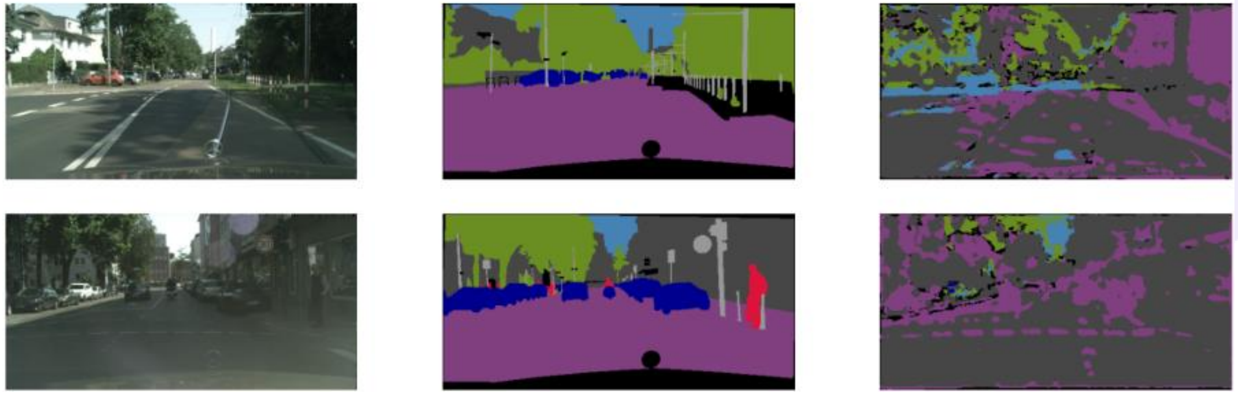




model	backbone	loss_function	val_iou	val_f_score	training_time	augmentation
baseline		dice+focal	0.388	0.490	259.22 min	No
unet	resnet34	dice+focal	0.657	0.702	79.74 min	No

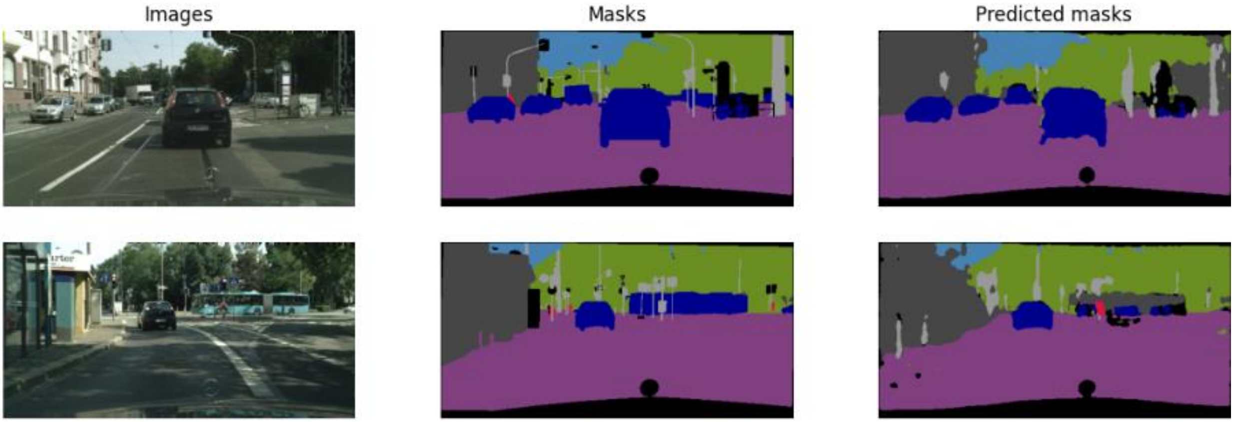
Architecture U-net – VGG16

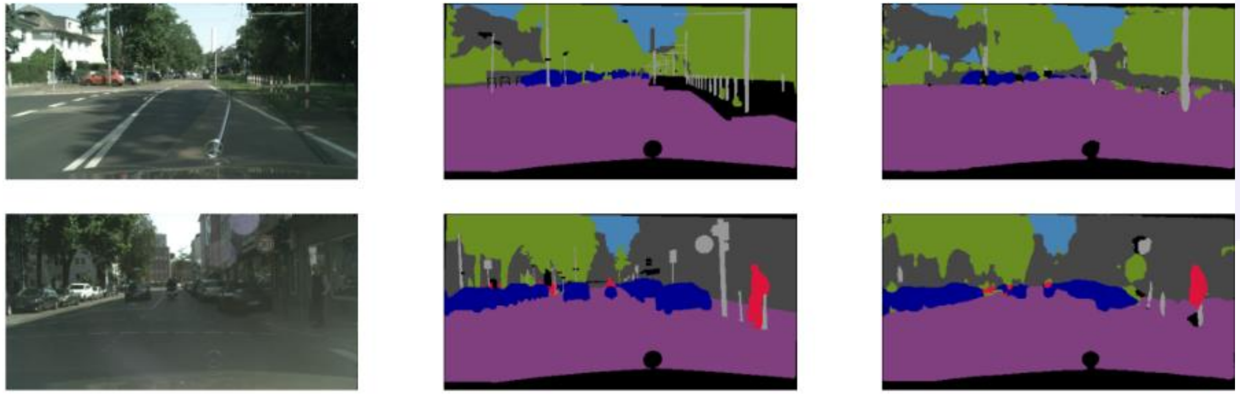




model	backbone	loss_function	val_iou	val_f_score	training_time	augmentation
baseline		dice+focal	0.388	0.490	259.22 min	No
unet	resnet34	dice+focal	0.657	0.702	79.74 min	No
unet	vgg16	dice+focal	0.302	0.347	179.79 min	No

avec Data Augmentation





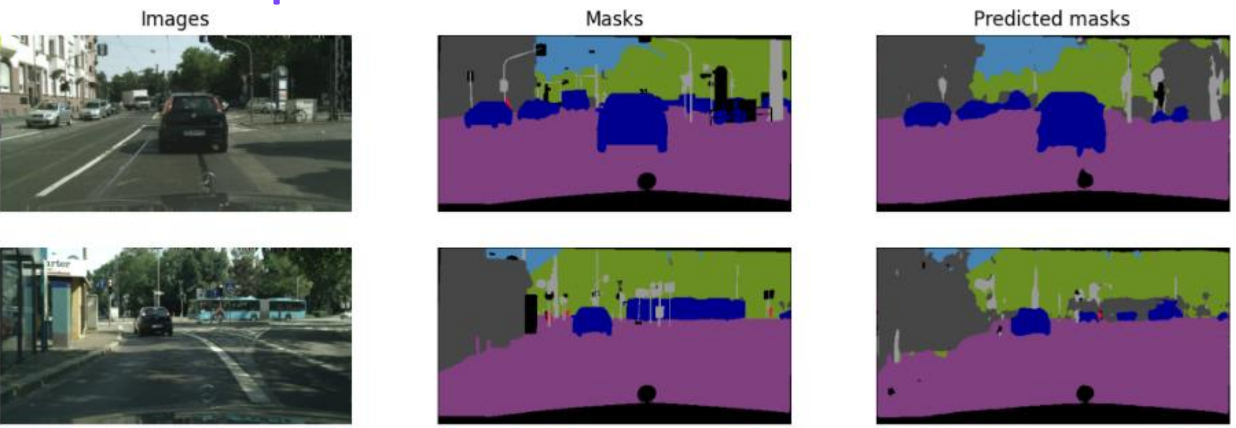
model	backbone	loss_function	val_iou	val_f_score	training_time	augmentation
baseline		dice+focal	0.388	0.490	259.22 min	No
unet	resnet34	dice+focal	0.657	0.702	79.74 min	No
unet	vgg16	dice+focal	0.302	0.347	179.79 min	No
unet	resnet34	dice+focal	0.658	0.722	157.30 min	Yes

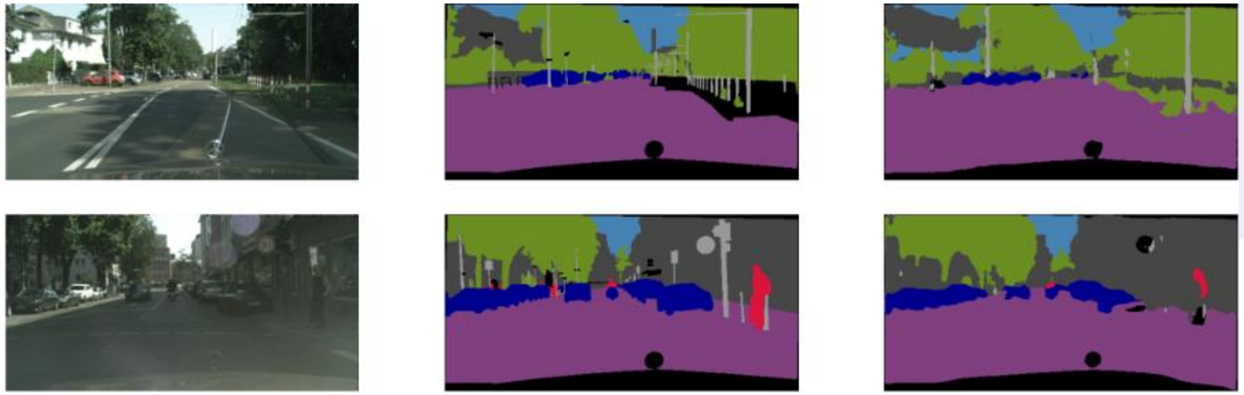
Optimisation fonction de perte : `class_weights`

- Loss = Focal loss + Dice loss
- Idée : utiliser l'IoU par classe

	Category	IoU	class_weights
0	void	0.768	0.110
1	flat	0.976	0.086
2	construction	0.921	0.092
3	object	0.359	0.236
4	nature	0.854	0.099
5	sky	0.925	0.091
6	human	0.506	0.167
7	vehicle	0.715	0.118

Optimisation fonction de perte : `class_weights`





model	backbone	loss_function	val_iou	val_f_score	training_time	augmentation
baseline		dice+focal	0.388	0.490	259.22 min	No
unet	resnet34	dice+focal	0.657	0.702	79.74 min	No
unet	vgg16	dice+focal	0.302	0.347	179.79 min	No
unet	resnet34	dice+focal	0.658	0.722	157.30 min	Yes
unet	resnet34	dice_CW+focal	0.664	0.732	147.38 min	Yes

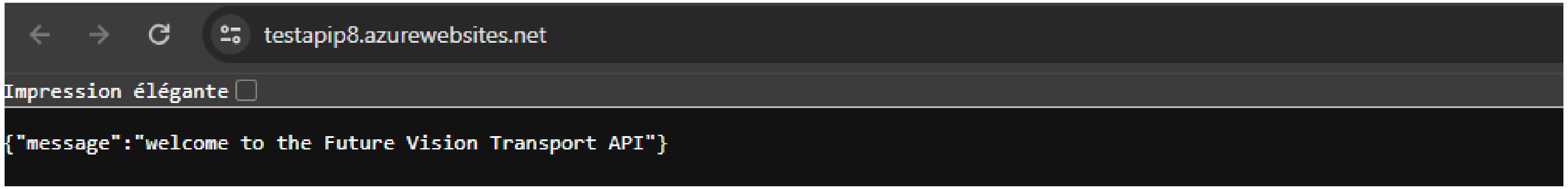
PARTIE 4 – API ET UI

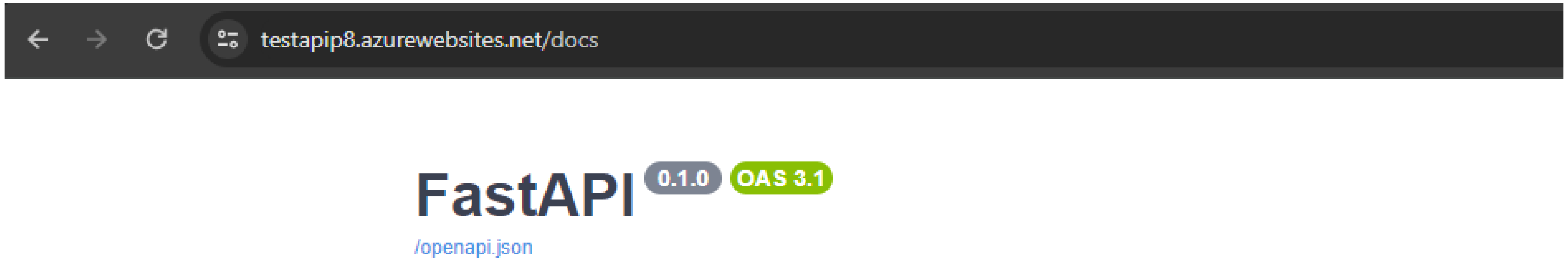
API

- entraîner modèle sur plus de données
- enregistrer TensorFlow Lite
- `main.py` FastAPI
- `Dockerfile` docker
- Construire image docker docker
- *resource group*
- *appservice plan*
- *webapp*
- Déploiement continu + GitHub Actions

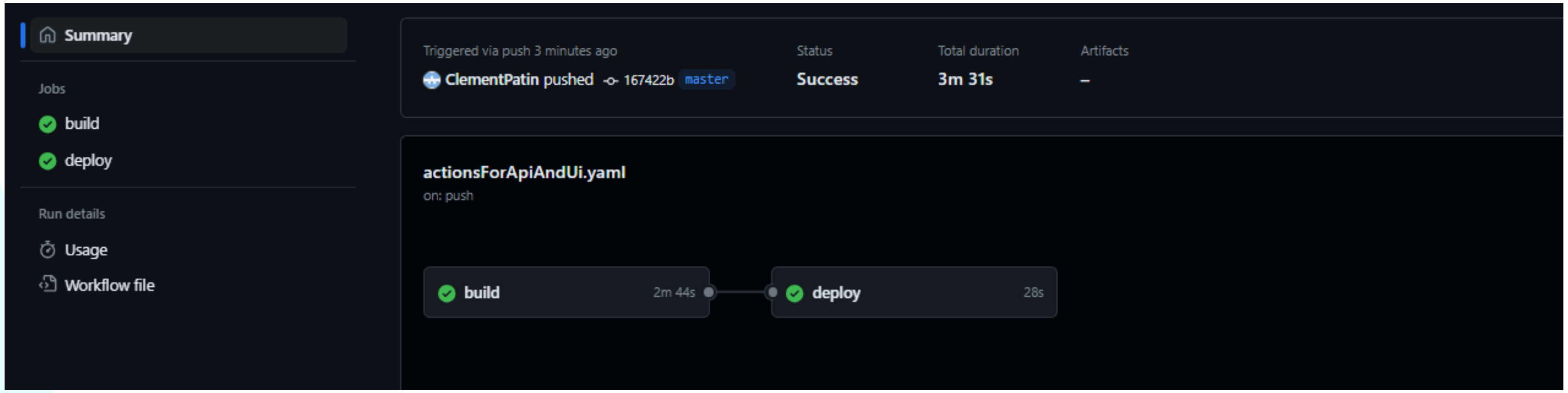
API

API déployée :

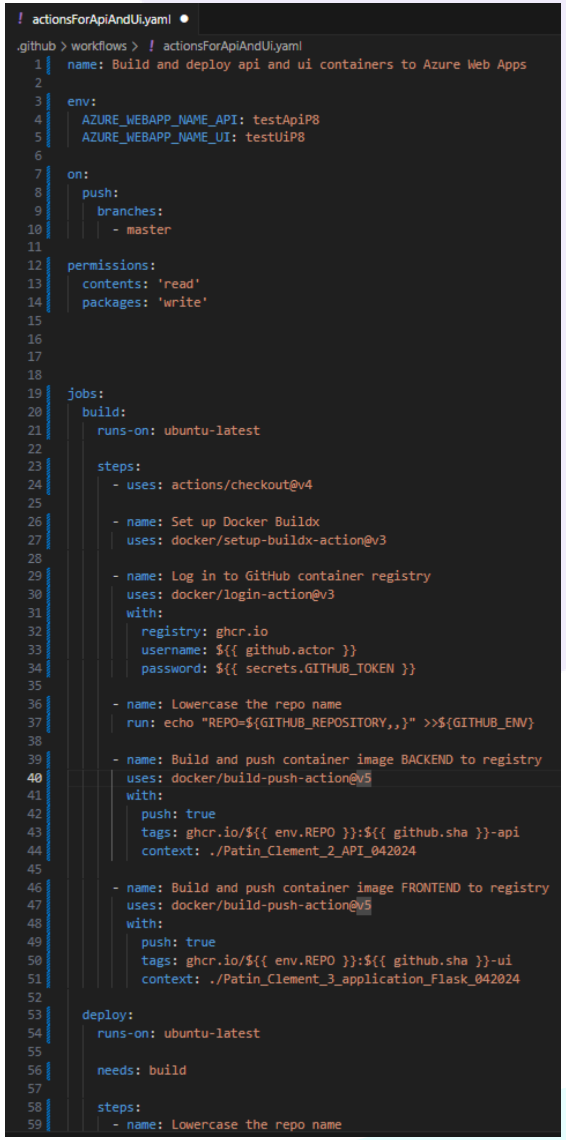




CI/CD:



Fichier YAML GitHub Actions :


actionsForApiAndUi.yaml

.github > workflows > actionsForApiAndUi.yaml

```yaml
name: Build and deploy api and ui containers to Azure Web Apps

env:
  AZURE_WEBAPP_NAME_API: testApiP8
  AZURE_WEBAPP_NAME_UI: testUiP8

on:
  push:
    branches:
      - master

permissions:
  contents: 'read'
  packages: 'write'



jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Log in to GitHub container registry
        uses: docker/login-action@v3
        with:
          registry: ghcr.io
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Lowercase the repo name
        run: echo "REPO=${GITHUB_REPOSITORY,,}" >>${GITHUB_ENV}

      - name: Build and push container image BACKEND to registry
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ghcr.io/${{ env.REPO }}:${{ github.sha }}-api
          context: ./Patin_Clement_2_API_042024

      - name: Build and push container image FRONTEND to registry
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ghcr.io/${{ env.REPO }}:${{ github.sha }}-ui
          context: ./Patin_Clement_3_application_Flask_042024

  deploy:
    runs-on: ubuntu-latest

    needs: build

    steps:
      - name: Lowercase the repo name
```

UI

- enregistrer 10 images et 10 masks dans `/static`
- `app.py` / `index.html` / `result.html` Flask
- `Dockerfile` docker
- Construire image docker docker
- *resource group*
- *appservice plan*
- *webapp*
- Déploiement continu + GitHub Actions

UI

app Flask :

```python
app.py

Patin_Clement_3_application_Flask_042024 > app.py > ...

app = Flask(__name__)

images_names = [str(i+1) for i in range(10)]  # List of image names

API_URL = "https://testapip8.azurewebsites.net"

@app.route("/")
def index():
    return render_template("index.html", images_names=images_names)

@app.route("/show_image/<image_name>")
def show_image(image_name):
    # Validate image name
    if image_name not in images_names:
        return "Invalid image name!"

    return render_template("result.html", images_names=images_names)

@app.route("/predict/<image_name>")
def predict(image_name):
    # check if images are in images
    if image_name not in images_names:
        return "Invalid image name!"

    # build url for this image
    image_path = url_for("static", filename="test_images/"+image_name+".png")
    mask_path = url_for("static", filename="test_masks/"+image_name+".png")

    # request API
    headers = {"accept" : "application/json"}
    files=[
        ('img',(image_name+".png", open("static/test_images/"+image_name+".png", 'r
    ]
    response = requests.post(url = API_URL+"/predict", headers=headers, files=files

    # extract image array from response
    predicted_mask = json.loads(response.json()["mask"])
```

CONCLUSION

merci