Future Vision Transport

DIFFERENTS MODELES DE SEGMENTATION SEMANTIQUE

Moussa Kibaly - INGENIEUR IA

Nous allons aborder dans cet article les différentes architectures choisies pour la conception d'un modèle de segmentation sémantique à partir du jeu de données Cityscape. Le modèle finale choisi a été développé et déployé sur Azure Machine Learning de Microsoft afin d'être utilisé par le service Web App :

- modèle SVM de référence
- modèle Unet
- Feature Pyramid Network (FPN)
- architecture VGG16-Unet

Le jeu de données Cityscape est composé de 2975 images d'entrainement et les labels associés, 500 images et labels de validation, 1525 images et labels de test.

1. SVM (Support Vector Machine)

Ce modèle est adapté pour les petits jeux de données et accepte en entrée des données de dimension 2.
Les images en couleur du jeu d'entrainement sont convertis en gris et convertis en vecteur. La classe GridSearchCV est utilisée afin d'obtenir les meilleurs paramètres du modèle parmi le paramètre C, gamma et le noyau (gaussien ou polynomial).

```
CPU times: user 30min 51s, sys: 16.7 s, total: 31min 8s
Wall time: 31min
GridSearchCV(cv=3, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=800,
                           probability=True, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [0.1, 1, 10, 100],
                         'gamma': [0.0001, 0.001, 0.1, 1],
                         'kernel': ['rbf', 'poly']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

L'indication CPU TIME indique le temps d'exécution du modèle SVM.

PERFORMANCE DU MODELE

- **Accuracy**

```
Accuracy  0.683891968872387
```

- **Rapport de la classification**

```
Classification report                    precision    recall

           3       0.00      0.00      0.00      3211
          11       0.43      0.00      0.00      1301
          20       0.00      0.00      0.00      1509
          21       0.70      0.98      0.82     12495
          23       0.54      1.00      0.70      1145

    accuracy                           0.68     19661
   macro avg       0.33      0.40      0.31     19661
weighted avg       0.51      0.68      0.56     19661
```

- **Paramètres optimales**

```
The optimal parameters are {'C': 100, 'gamma': 1, 'kernel': 'poly'} with a score of 0.6872
```

- **F1-score et F50-mesure**

```
F-mesure                    : 0.5616
F50-mesure                  : 0.6838
```

- **Indice de Jaccard**

```
Indice de Jaccard  0.472232710987850 97
```

- **Fonction de perte**

```
Log Loss  1.1072729869195634
```

2. Modèle Unet

Ce modèle a une architecture encodeur-décodeur.
L' encodeur est composé d'empilement de couches complexes contenant chacune 2 couches de convolution 2D utilisant la fonction d'activation ReLu suivies par un max pooling. Le principe de la partie encodeur est de produire des tenseurs de basse dimension contenant toutes les informations spatiales de l'image.
Le décodeur permet de récupérer les tenseurs de basse dimension afin de produire en sortie des cartes de segmentation (segmentation sémantique).
L'augmentation de dimension des tenseurs se fait par une couche Upsampling combinée avec une couche de convolution + ReLu et de max pooling.
Des sauts de connexion sont faites entre les sorties des couches intermédiaires de convolution de l'encodeur et l'entrée de ces mêmes couches du décodeur.
Ces sauts de connexion permettent de remédier à la perte d'information le long des couches empilées.

Nous avons optimisé les hyper-paramètres tels que le taux d'apprentissage du modèle et la fonction d'activation :

```
The hyperparameter search is complete.
 The optimal activaion function is relu And the optimal learning rate for the optimizer is 0.001.
```

Un taux d'apprentissage de 0.01 sur l'algorithme Adam et l'utilisation de la fonction Relu semblent apporter une meilleur performance du modèle Unet.

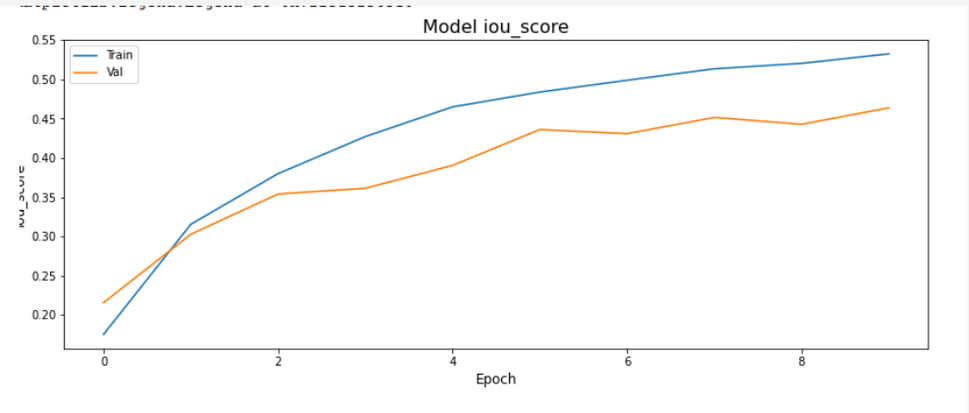
```
# DOWNSAMPLING LAYERS
conv_1 = Conv2D(32, (3, 3), activation='relu', padding='same', kernel_initializer = 'he_normal')(input_
conv_1 = Dropout(0.2)(conv_1)
conv_1 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv_1)
pool_1 = MaxPooling2D((2, 2))(conv_1)

conv_2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(pool_1)
conv_2 = Dropout(0.2)(conv_2)
conv_2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv_2)
pool_2 = MaxPooling2D((2, 2))(conv_2)

conv_3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(pool_2)
conv_3 = Dropout(0.2)(conv_3)
conv_3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv_3)

# UPSAMPLING LAYERS
up_1 = concatenate([UpSampling2D((2, 2))(conv_3), conv_2], axis=-1)
conv_4 = Conv2D(32, (3, 3), activation='relu', padding='same')(up_1)
conv_4 = Dropout(0.2)(conv_4)
conv_4 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv_4)

up_2 = concatenate([UpSampling2D((2, 2))(conv_4), conv_1], axis=-1)
conv_5 = Conv2D(32, (3, 3), activation='relu', padding='same')(up_2)
conv_5 = Dropout(0.2)(conv_5)
conv_5 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv_5)

segm_X = Conv2D(INPUT_CLASS_NB, (1, 1), activation='softmax', name='seg')(conv_5)

unet_model = Model(inputs=[input_0], outputs=[segm_X])
```

PERFORMANCE

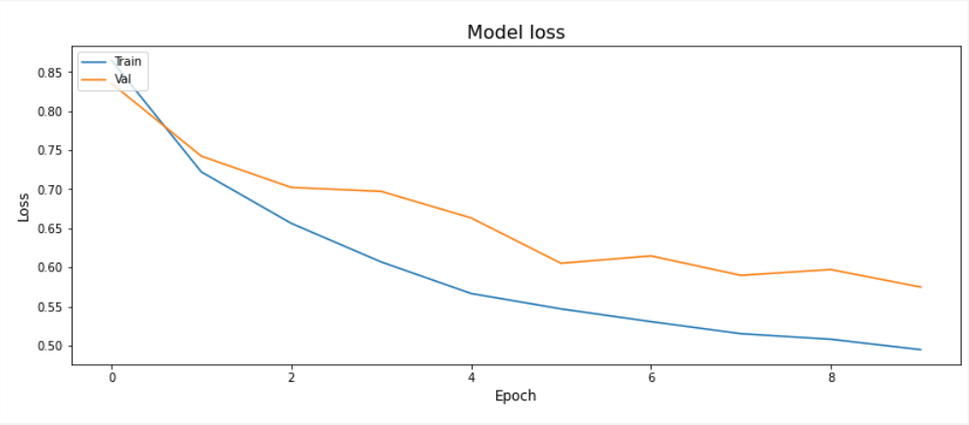
On sauvegarde le modèle qui a le meilleur score de IoU de validation. Le score Intersection over Union sert à mesurer l'adéquation entre l'image segmentée et l'image réelle, elle est plus adaptée pour la segmentation sémantique que ne peut l'être la précision (accuracy).

De plus, le modèle doit minimiser la fonction de perte choisie categorical_focal_jaccard_loss et maximiser le score IOU et Accuracy sur les jeux de données d'entrainement et de validation.

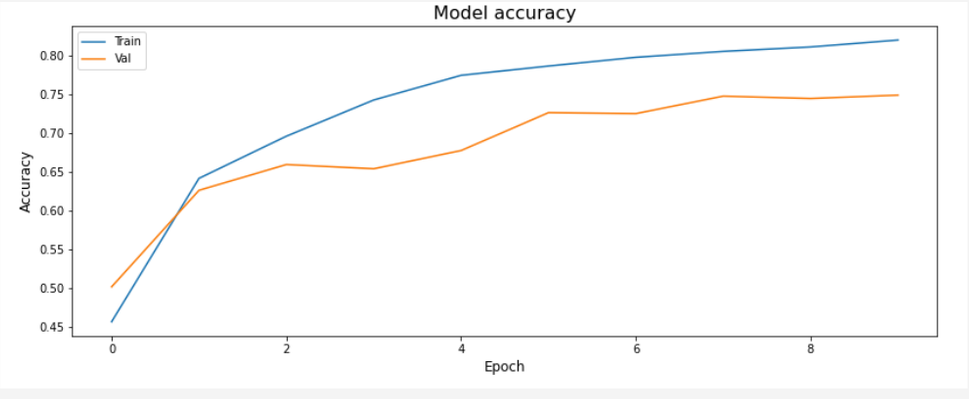
- **IOU_SCORE de 53.24%**



- **Fonction de perte de 49.46%**



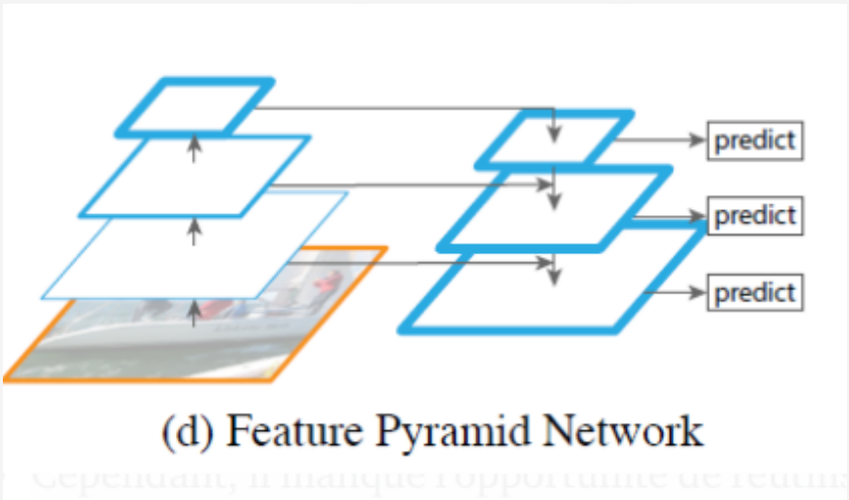
- **Accuracy de 81.99%**



Les métriques de performance du modèle ont pu être amélioré par le biais de la technique d'augmentation d'images en utilisant des packages tels que Albumentations ou Imgaug.
Cela consiste à générer de nouvelles images sur la base des images d'origine en effectuant des rotations, des zooms, des translations ou applications de filtres et autres techniques d'image.
Sans l'augmentation d'image, le modèle a généré un **IOU_SCORE de 32.84%, une fonction de perte de 71.52% et un Accuracy de 59.88%.**

3. Feature Pyramid Network (FPN)

Ce modèle est composé de 23 939 025 paramètres.



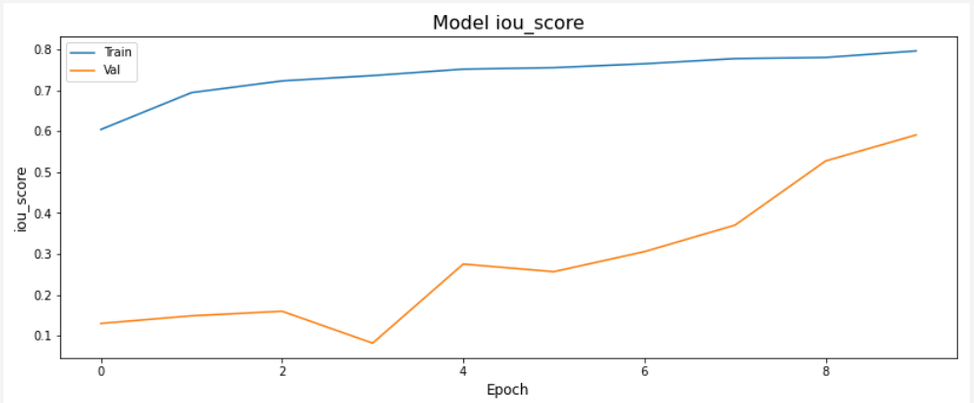
(d) Feature Pyramid Network

```
final_stage_conv (Conv2D)       (None, None, None, 1 589824
final_stage_bn (BatchNormalizat (None, None, None, 1 512
final_stage_relu (Activation)   (None, None, None, 1 0
final_upsampling (UpSampling2D) (None, None, None, 1 0
head_conv (Conv2D)              (None, None, None, 8 9224
softmax (Activation)            (None, None, None, 8 0
==========================================================
Total params: 23,939,025
Trainable params: 23,921,355
Non-trainable params: 17,670
```

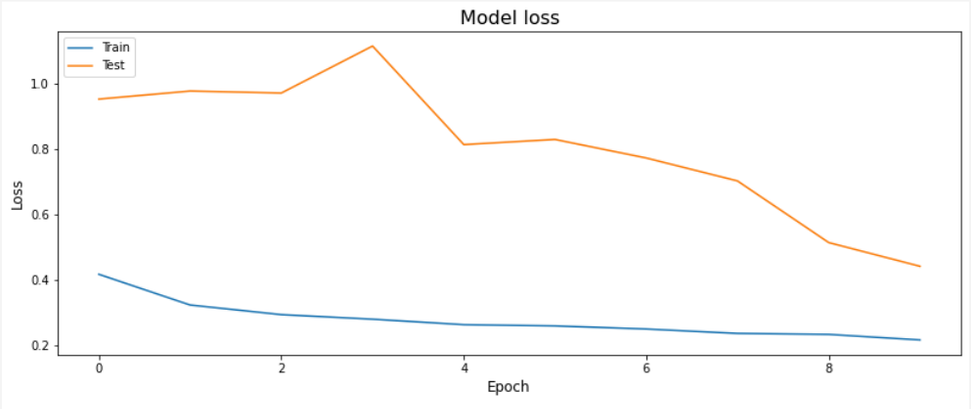
PERFORMANCE

Le modèle combine des caractéristiques à faible résolution et sémantiquement fortes avec des caractéristiques à haute résolution et sémantiquement faibles via une voie descendante et des connexions latérales.

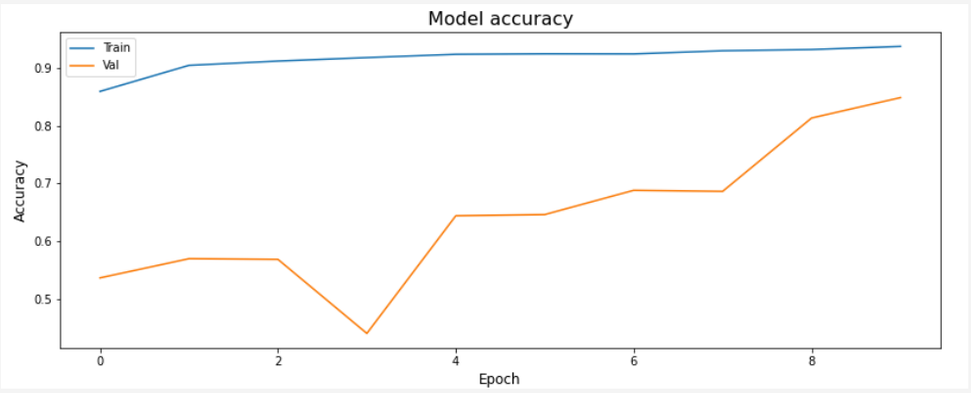
- **IOU_SCORE de 79.59%**



- **Fonction de perte de 21.52%**



- **Accuracy de 93.77%**

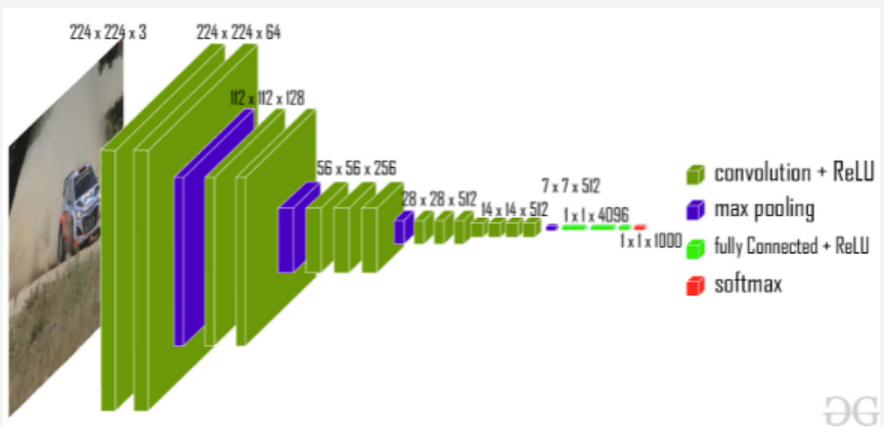


4. Modèle combiné VGG16-Unet

Ce modèle est composé de 23 753 288 paramètres.

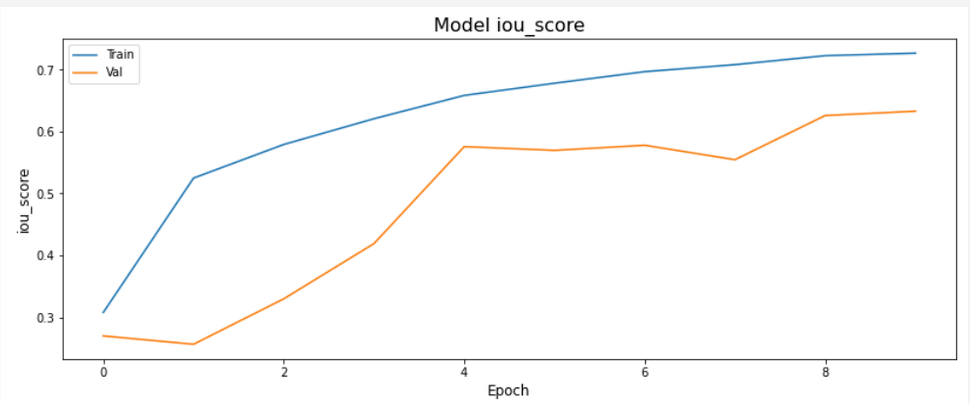
```
decoder_stage4a_bn (BatchNormal (None, None, None, 1 64          decoder_stage4a_conv[0][0]
__________________________________________________________________________________________________
decoder_stage4a_relu (Activatio (None, None, None, 1 0           decoder_stage4a_bn[0][0]
__________________________________________________________________________________________________
decoder_stage4b_conv (Conv2D)   (None, None, None, 1 2304        decoder_stage4a_relu[0][0]
__________________________________________________________________________________________________
decoder_stage4b_bn (BatchNormal (None, None, None, 1 64          decoder_stage4b_conv[0][0]
__________________________________________________________________________________________________
decoder_stage4b_relu (Activatio (None, None, None, 1 0           decoder_stage4b_bn[0][0]
__________________________________________________________________________________________________
final_conv (Conv2D)             (None, None, None, 8 1160        decoder_stage4b_relu[0][0]
__________________________________________________________________________________________________
softmax (Activation)            (None, None, None, 8 0           final_conv[0][0]
==================================================================================================
Total params: 23,753,288
Trainable params: 23,749,256
Non-trainable params: 4,032
__________________________________________________________________________________________________
```

Ce modèle Unet comprend un encodeur VGG16. VGG16 est l'architecture d'un réseau de neurone convolutionnel utilisé pour la classification et la détection d'images.

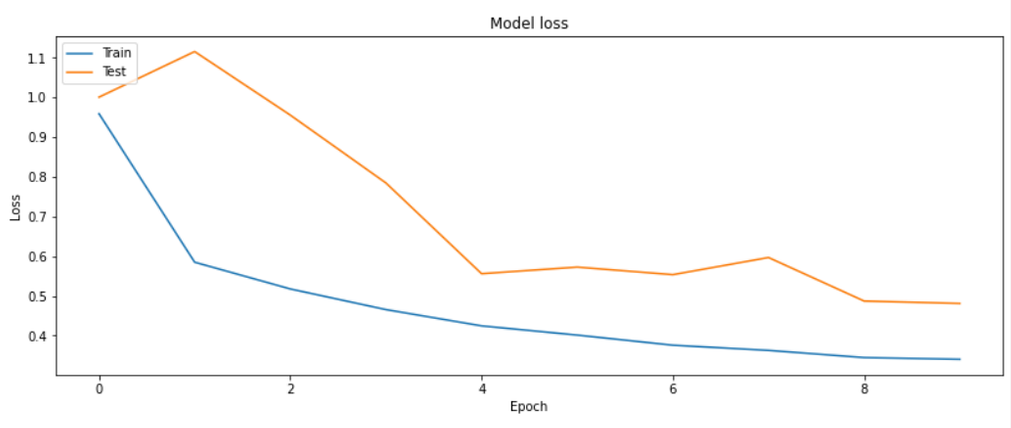


Evaluation des performances

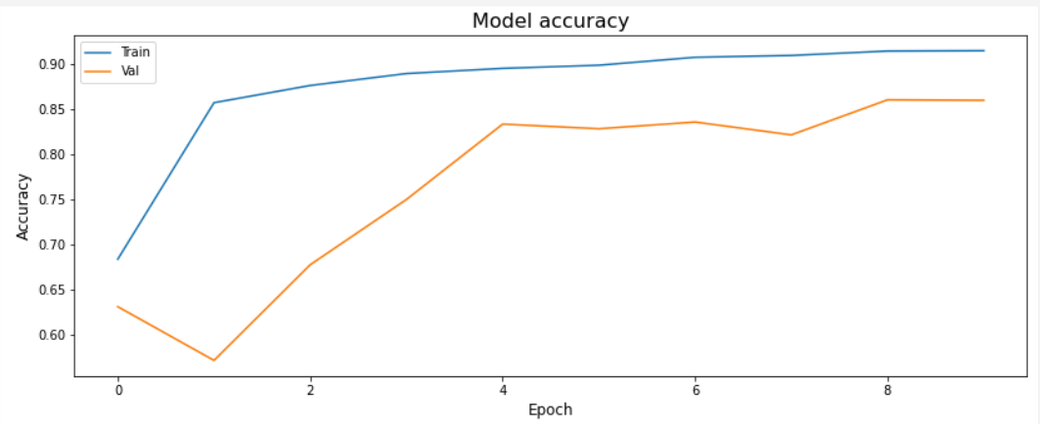
- **IOU_SCORE de 72.68%**



- **Fonction de perte de 34.04%**



- **Accuracy de 91.49%**



CONCLUSION

Le modèle Unet a été choisie en raison de sa simplicité d'implémentation et les résultats satisfaisants obtenus par le biais de l'augmentation d'images.