Modélisation avancée

- Choix de modèles pré-entrainés
- Optimisation : EarlyStopping , Keras Tuner

Classification:

- Binaire: Sain/Malade
- Multiclasse : 4 classes : Normal / Viral_Pneumonia / Lung_Opacity / COVID

Les modèles sélectionnés

- VGG16
- VGG19
- ResNet50
- EfficientNetB0

Nous avons sélectionnés ces modèles pour les raisons suivantes:

- Utilisation des mêmes dimensions des images
- Dimensions nécessaires sont inférieures aux dimensions des images brutes
  Inconvénient:
- Besoin de conserver les images en RGB, contrairement à notre approche initiale de remise en Niveau de Gris de toutes les images

Les modèles sélectionnés par classification:

- Binaire Sain / Malade: VGG16, VGG19
- Multiclasse : ResNet50 et EfficientNetB0

Les données

- 600 images par classe sélectionnés aléatoirement
- Dimensions par défaut des modèles : 224x224x3 (RGB)

Les techniques d'optimisation utilisées

EfficientNetB0:

- EarlyStopping : EfficientNetB0
- Keras Tuning RandomSearch : EfficientNetB0 / VGG16 et VGG19

- Dégel de couches pré-entrainées : EfficientNetB0 / VGG16 et VGG19

Les données

- 600 images par classe sélectionnés aléatoirement
- Dimensions standards des modèles

Métriques

- F1 Score
- Recall
- Restitution: Matrice de confusion et Rapport de classification

Interprétabilité:

GradCAM

Cas 1 - Binaire : Sain / Malade

1. Taille = 28
2. Images = 600 images par classe

- Sain = Normal
- Malade = 1/3 COVID, 1/3 Viral Pneumonia / 1/3 Lung Opacity

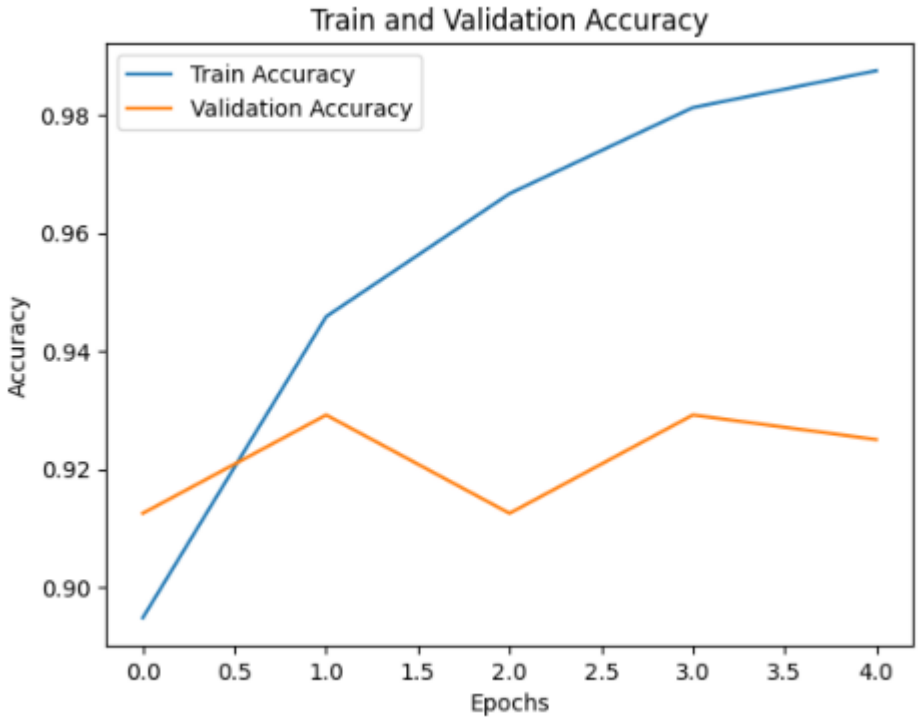
In [7]:
```
import matplotlib.pyplot as plt
from PIL import Image

def load_and_display_image(image_path):
    img = Image.open(image_path)
    plt.imshow(img)
    plt.axis('off')  # Cela enlève les axes
    plt.show()
```

VGG 16

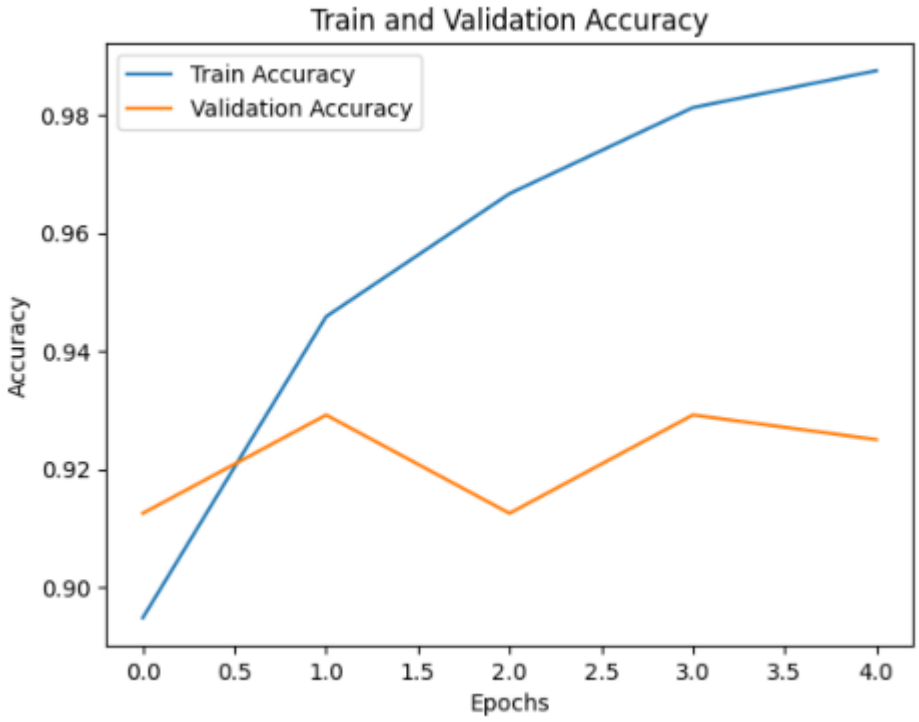
Courbe d'entrainement

VGG16 Sain/Malade - Précision

In [21]:
```
load_and_display_image("./metriques_courbes/vgg16_entrainement_accuracy.p
```

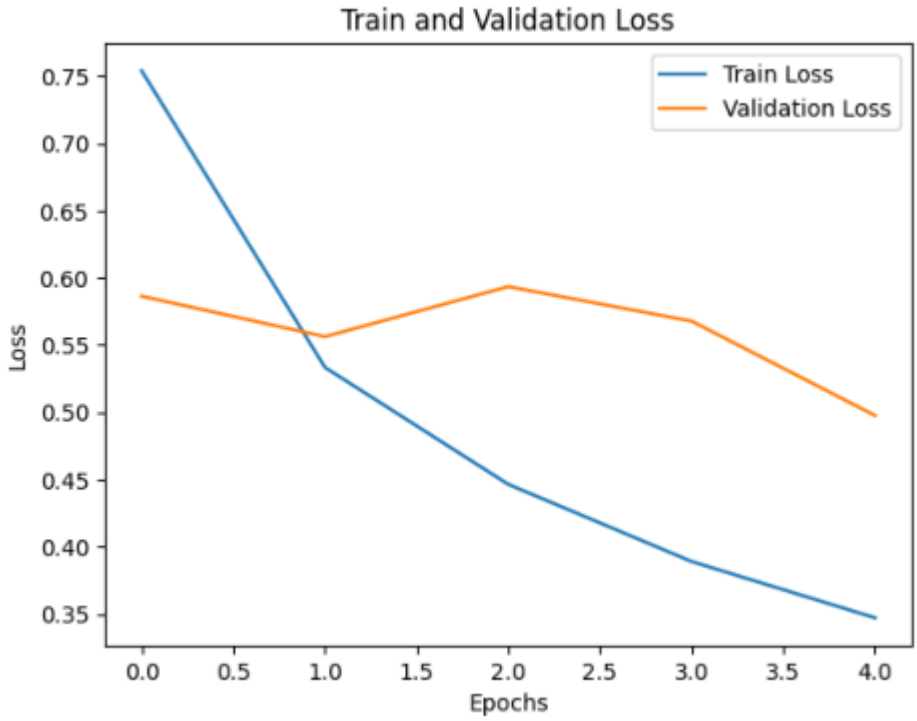


In [21]:
```
load_and_display_image("./metriques_courbes/vgg16_entrainement_accuracy.p
```



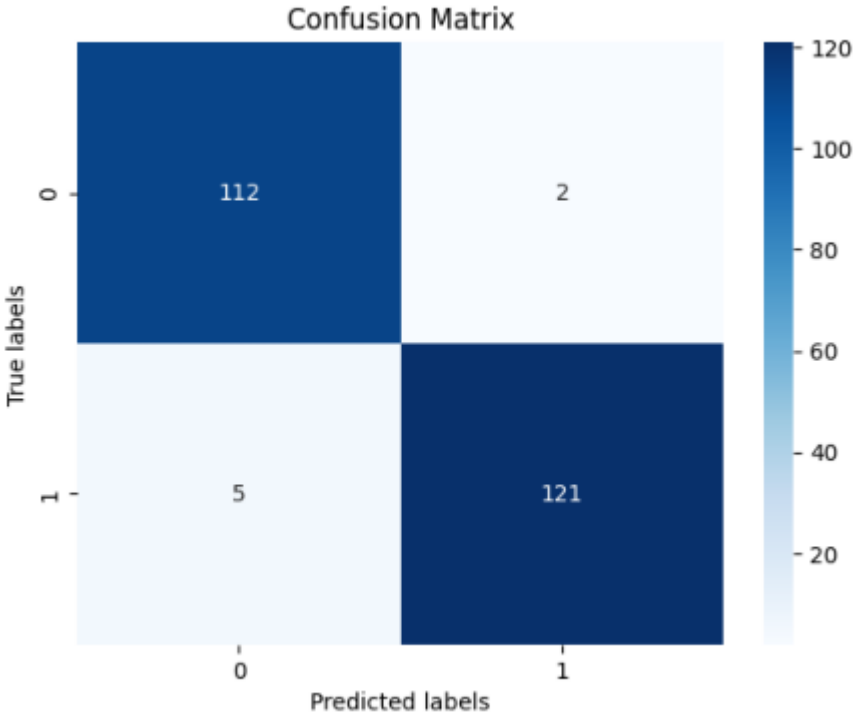
VGG16 Sain/Malade - Perte

In [22]:
```
load_and_display_image("./metriques_courbes/vgg16_entrainement_loss.png")
```



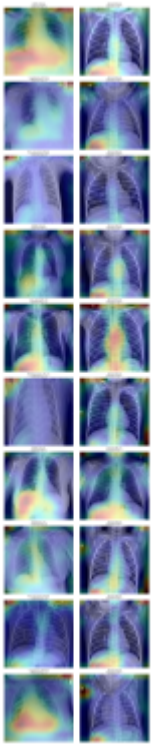
VGG16 Sain/Malade - Matrice de confusion

In [23]:
```
load_and_display_image("./metriques_courbes/VGG16_confusion_matrix.png")
```



VGG16 Sain/Malade - Gradcam

In [24]:
```
load_and_display_image("./interpretabilite_gradcam/VGG16_GradCam.png")
```



VGG19

Courbe d'entrainement

VGG19 Sain/Malade - Précision

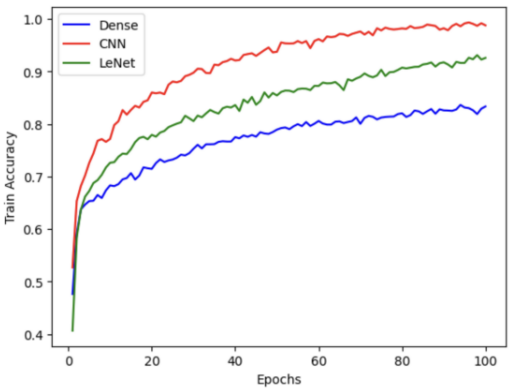
In [40]:
```
# tous les noms de fichiers
# sparse
mc_28_sparse_dense_accuracy=os.path.join(chemin_principal,'MC_28_Sparse_D
mc_28_sparse_train_accuracy=os.path.join(chemin_principal,'MC_28_Sparse_T
mc_28_sparse_dense_loss=os.path.join(chemin_principal,'MC_28_Sparse_Dense
mc_28_sparse_train_loss=os.path.join(chemin_principal,'MC_28_Sparse_Train
mc_28_sparse_cnn_accuracy=os.path.join(chemin_principal,'MC_28_Sparse_CNN
mc_28_sparse_lenet_accuracy=os.path.join(chemin_principal,'MC_28_Sparse_L
mc_28_sparse_val_accuracy=os.path.join(chemin_principal,'MC_28_Sparse_Val
mc_28_sparse_cnn_loss=os.path.join(chemin_principal,'MC_28_Sparse_CNN_Los
mc_28_sparse_lenet_loss=os.path.join(chemin_principal,'MC_28_Sparse_LeNet
mc_28_sparse_val_loss=os.path.join(chemin_principal,'MC_28_Sparse_Val_Los

# no sparse

mc_28_no_sparse_cnn_accuracy=os.path.join(chemin_principal,'MC_28_No_Spar
mc_28_no_sparse_lenet_accuracy=os.path.join(chemin_principal,'MC_28_No_Sp
mc_28_no_sparse_val_accuracy=os.path.join(chemin_principal,'MC_28_No_Spar
mc_28_no_sparse_cnn_loss=os.path.join(chemin_principal,'MC_28_No_Sparse_C
mc_28_no_sparse_lenet_loss=os.path.join(chemin_principal,'MC_28_No_Sparse
mc_28_no_sparse_val_loss=os.path.join(chemin_principal,'MC_28_No_Sparse_V
mc_28_no_sparse_dense_accuracy=os.path.join(chemin_principal,'MC_28_No_Sp
mc_28_no_sparse_train_accuracy=os.path.join(chemin_principal,'MC_28_No_Sp
mc_28_no_sparse_dense_loss=os.path.join(chemin_principal,'MC_28_No_Sparse
mc_28_no_sparse_train_loss=os.path.join(chemin_principal,'MC_28_No_Sparse
```

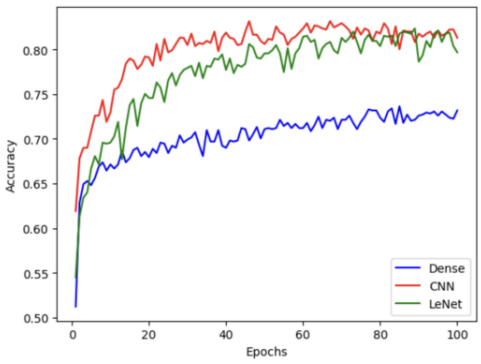
loss='sparse_categorical_crossentropy'

Accuracy 3 modèles

- Train accuracy
- Validation accuracy

In [41]:
```
# Charger les images
image1 = plt.imread(mc_28_sparse_train_accuracy)
image2 = plt.imread(mc_28_sparse_val_accuracy)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```



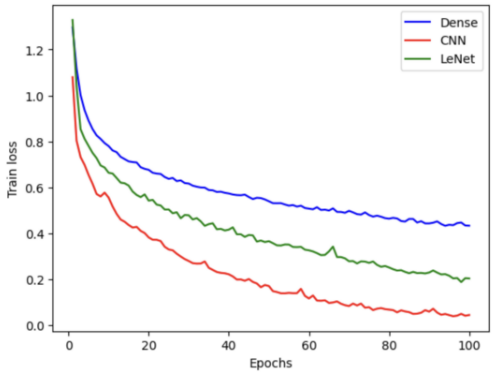


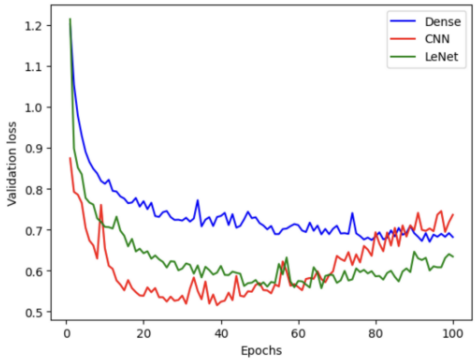
Loss 3 modèles

- Train loss
- Validation loss

In [42]:
```
# Charger les images
image1 = plt.imread(mc_28_sparse_train_loss)
image2 = plt.imread(mc_28_sparse_val_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
```

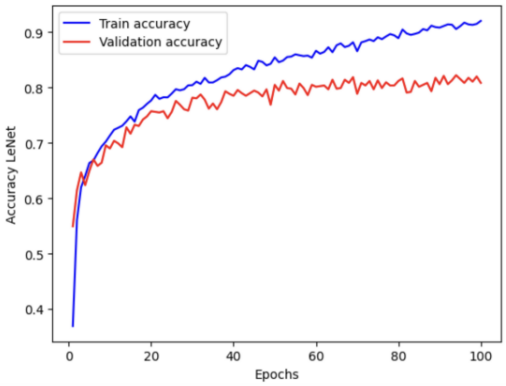
```
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```

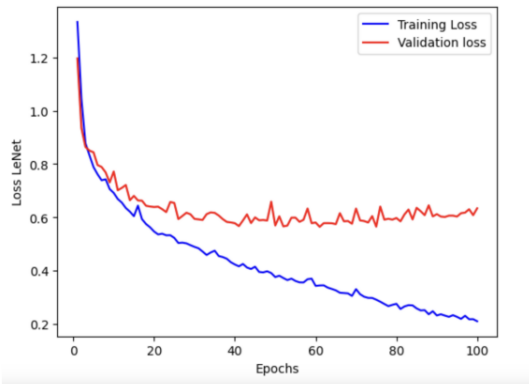




Accuracy LeNet / Loss LeNet

In [43]:
```
# Charger les images
image1 = plt.imread(mc_28_sparse_lenet_accuracy)
image2 = plt.imread(mc_28_sparse_lenet_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```

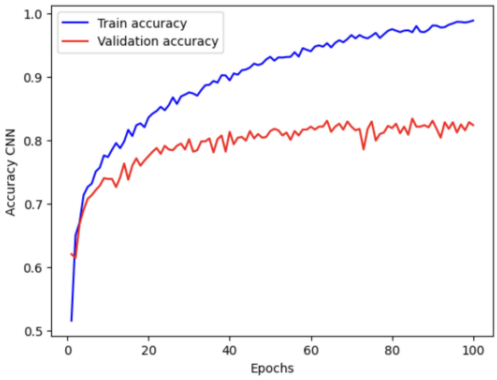


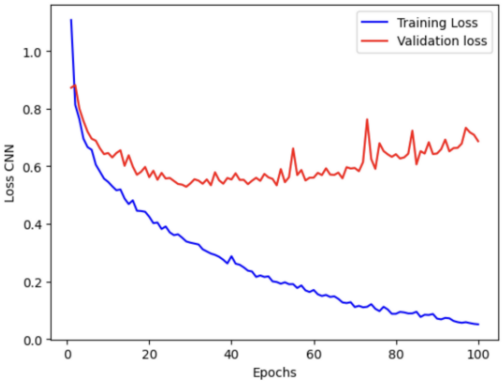


Accuracy CNN / Loss CNN

In [44]:
```
# Charger les images
image1 = plt.imread(mc_28_sparse_cnn_accuracy)
image2 = plt.imread(mc_28_sparse_cnn_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
```

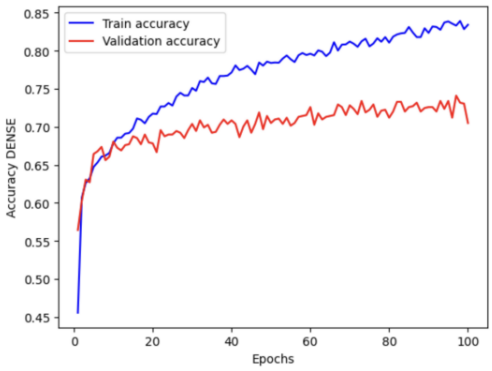
```
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```

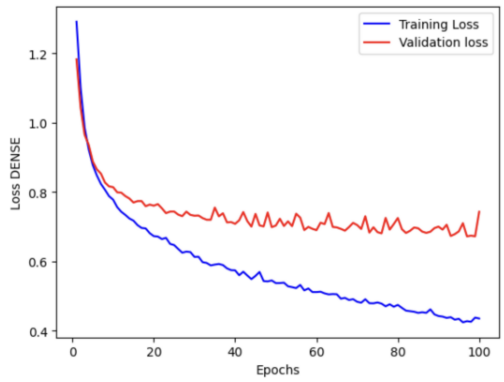




Accuracy DENSE / Loss DENSE

In [45]:
```
# Charger les images
image1 = plt.imread(mc_28_sparse_dense_accuracy)
image2 = plt.imread(mc_28_sparse_dense_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```



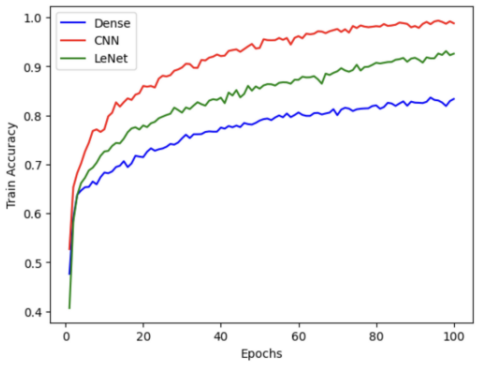


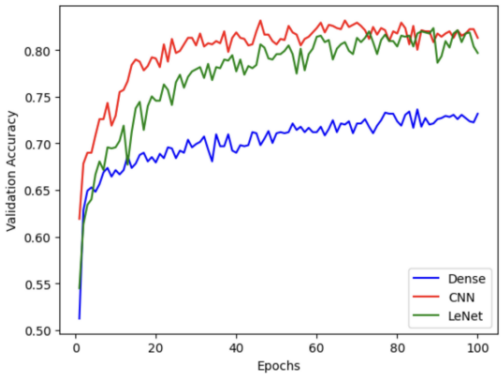
loss='categorical_crossentropy'

Accuracy 3 modèles

- Train accuracy
- Validation accuracy

In [46]:
```
# Charger les images
image1 = plt.imread(mc_28_no_sparse_train_accuracy)
image2 = plt.imread(mc_28_no_sparse_val_accuracy)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```

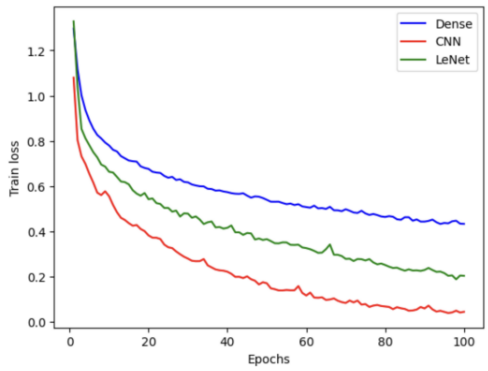


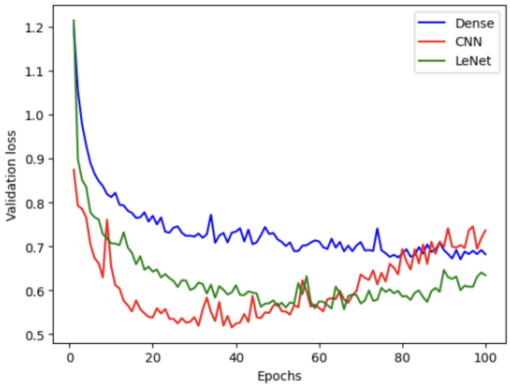


Loss 3 modèles

- Train loss
- Validation loss

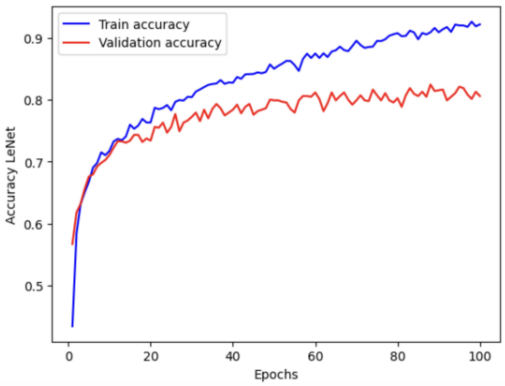
In [47]:
```
# Charger les images
image1 = plt.imread(mc_28_no_sparse_train_loss)
image2 = plt.imread(mc_28_no_sparse_val_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```

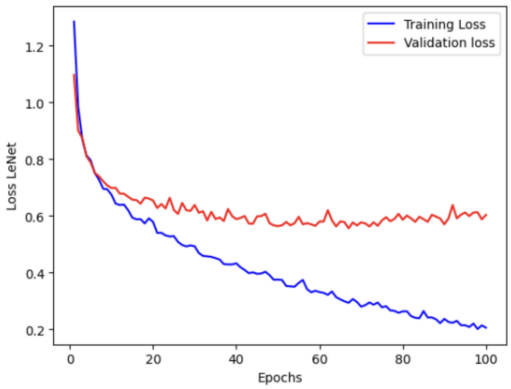




Accuracy LeNet / Loss LeNet

In [48]:
```
# Charger les images
image1 = plt.imread(mc_28_no_sparse_lenet_accuracy)
image2 = plt.imread(mc_28_no_sparse_lenet_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```

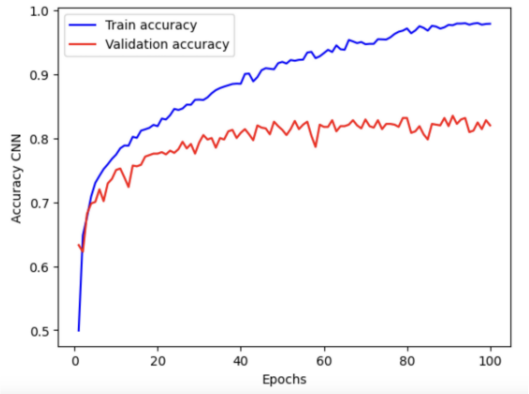


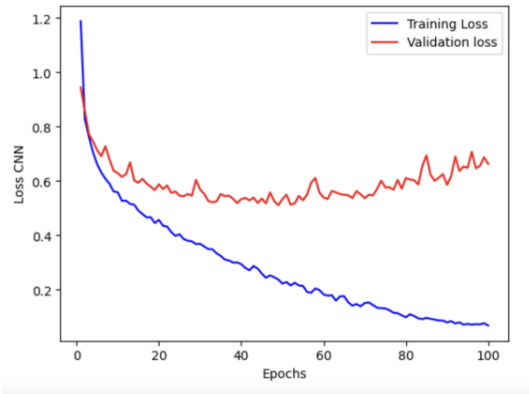


Accuracy CNN / Loss CNN

In [49]:
```
# Charger les images
image1 = plt.imread(mc_28_no_sparse_cnn_accuracy)
image2 = plt.imread(mc_28_no_sparse_cnn_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image
```

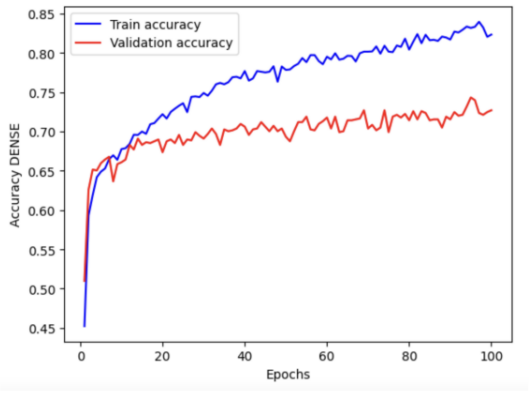
```
# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```

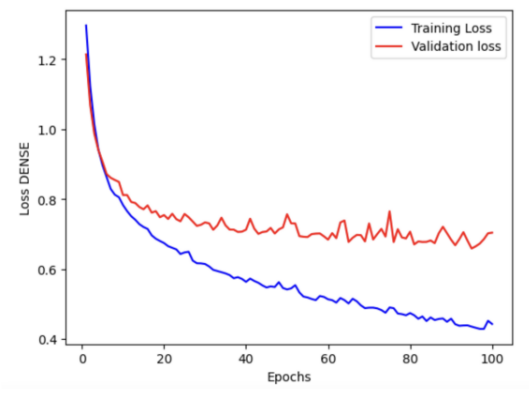




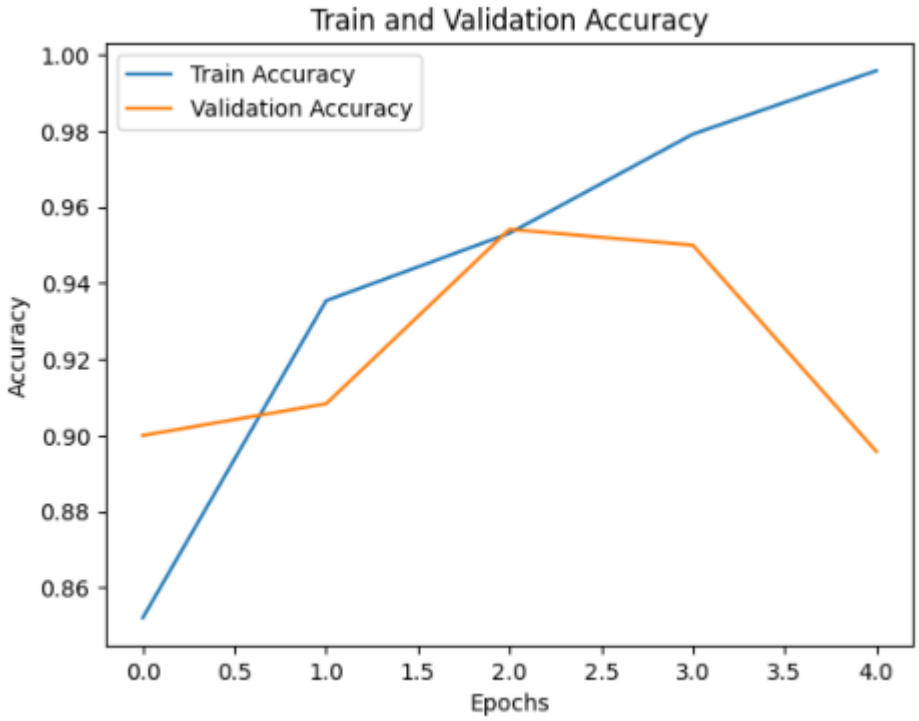
Accuracy DENSE / Loss DENSE

In [50]:
```
# Charger les images
image1 = plt.imread(mc_28_no_sparse_dense_accuracy)
image2 = plt.imread(mc_28_no_sparse_dense_loss)

# Créer une figure et un ensemble d'axes
fig, axes = plt.subplots(1, 2, figsize=(16, 8))  # 1 ligne, 2 colonnes

# Afficher la première image
axes[0].imshow(image1)
axes[0].axis('off')  # Masquer les axes pour la première image

# Afficher la seconde image
axes[1].imshow(image2)
axes[1].axis('off')  # Masquer les axes pour la seconde image

plt.show()
```



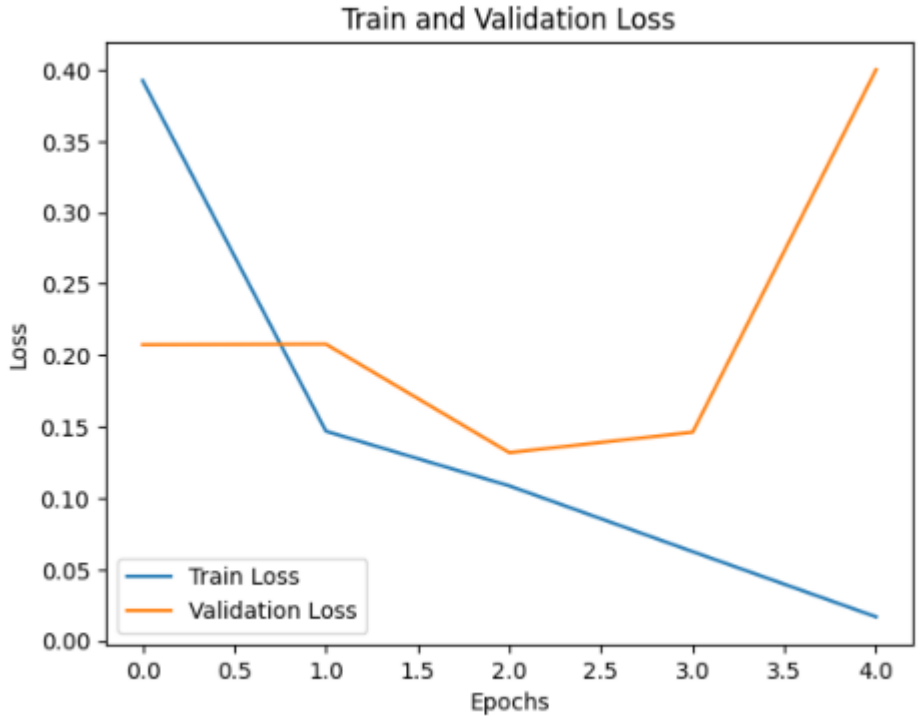


In [25]:
```
load_and_display_image("./metriques_courbes/vgg19_entrainement_accuracy.p
```



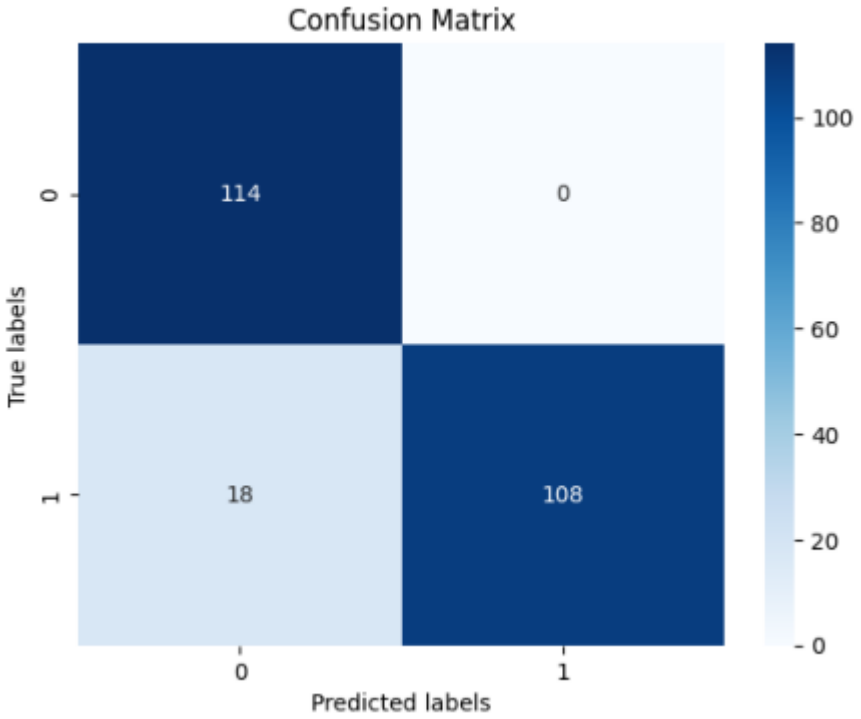
VGG19 Sain/Malade - Perte

In [26]:
```
load_and_display_image("./metriques_courbes/vgg19_entrainement_loss.png")
```



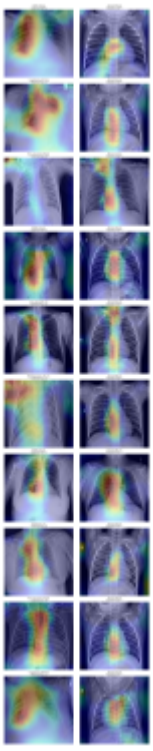
VGG19 Sain/Malade - Matrice de confusion

In [27]:
```
load_and_display_image("./metriques_courbes/VGG19_confusion_matrix.png")
```



VGG19 Sain/Malade - Gradcam

In [28]:
```
load_and_display_image("./interpretabilite_gradcam/VGG19_GradCam.png")
```



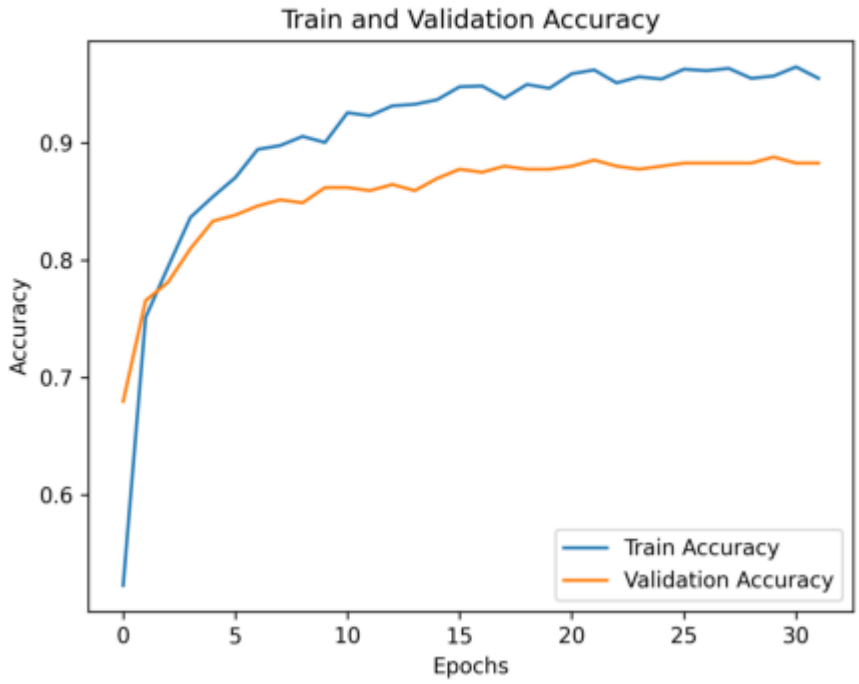
Cas 2 - Multiclasse

EfficientNet B0

Courbe d'entrainement

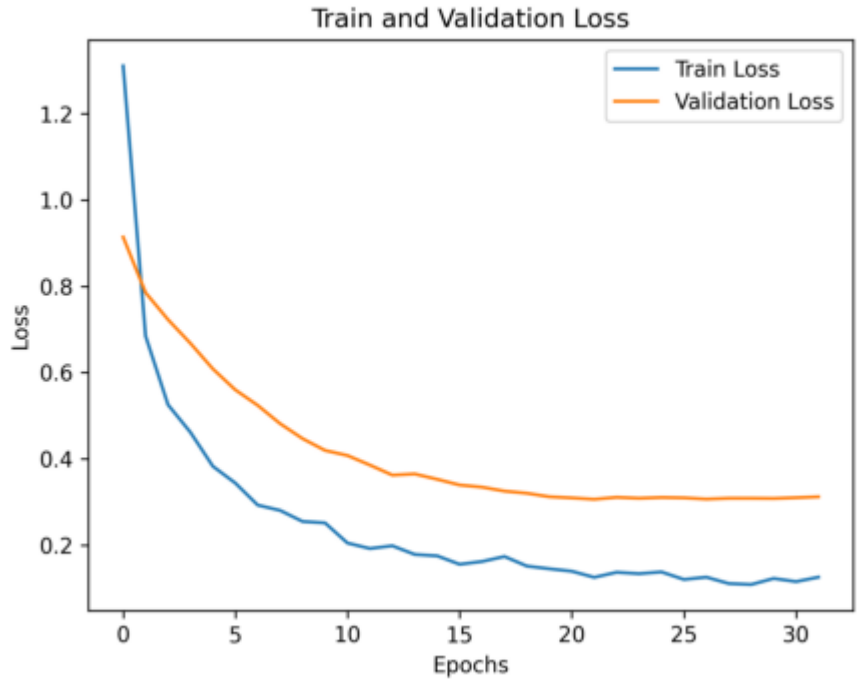
EfficientNetB0 Multiclasse - Précision

In [29]:
```
load_and_display_image("./metriques_courbes/MC_EfficientNetB0_train_valid
```



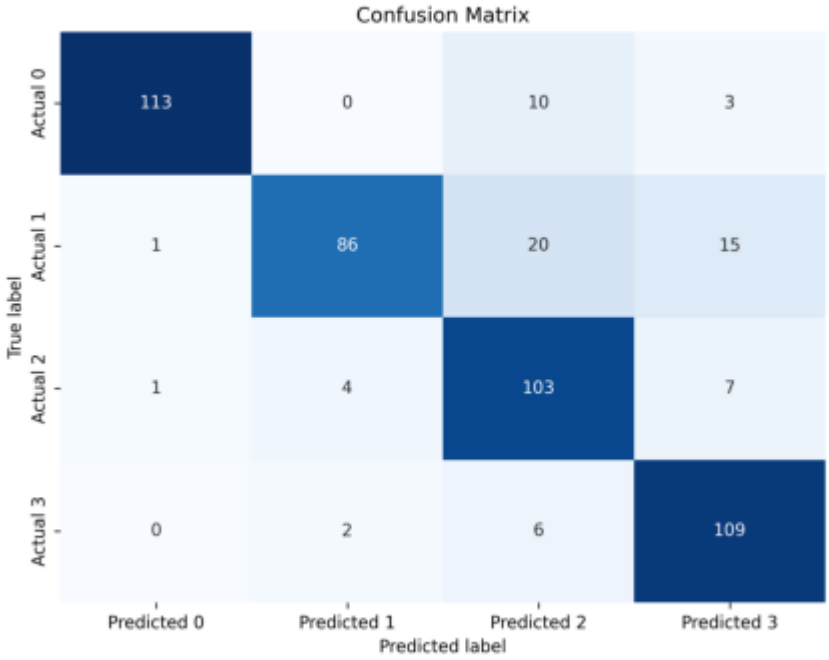
EfficientNetB0 Multiclasse - Perte

In [32]:
```
load_and_display_image("./metriques_courbes/MC_EfficientNetB0_train_valid
```



EfficientNetB0 Multiclasse - Matrice de confusion

In [34]:
```
load_and_display_image("./metriques_courbes/MC_EfficientNetB0_ba242671f16
```



EfficientNetB0 Multiclasse - Gradcam

In [35]:
```
### Non implémenté
```

ResNet 50

Courbe d'entrainement

ResNet 50 Multiclasse - Précision

In [ ]:
```
load_and_display_image("./metriques_courbes/ResNet50_train_validation_acc
```

ResNet 50 Multiclasse - Perte

In [ ]:
```
load_and_display_image("./metriques_courbes/ResNet50_train_validation_los
```