RAPPORT DE TESTS ET ANALYSES

Introduction

L'objectif de ce projet était de développer et de tester plusieurs modèles d'apprentissage automatique pour classifier des images de radiographies pulmonaires en deux classes : NORMAL et PNEUMONIA. Les données sont divisées en trois ensembles : entraînement (train), validation (val), et test (test).

Organisation des Données

Les données étaient organisées dans les répertoires suivants :

- train : Contient 5216 images réparties en 2 classes (NORMAL et PNEUMONIA).
- val : Contient 16 images réparties en 2 classes (NORMAL et PNEUMONIA).
- test : Contient 624 images réparties en 2 classes (NORMAL et PNEUMONIA).

Tests Réalisés

1. Affichage d'Exemples d'Images

Objectif : Visualiser un échantillon des images pour comprendre les données.

```
tabnine: test | explain | document | ask
def display_sample_images(folder, num_images=5):
    categories = ['NORMAL', 'PNEUMONIA']
    for category in categories:
        path = os.path.join(folder, category)
        images = os.listdir(path)
        for img_name in images[:num_images]:
            img_path = os.path.join(path, img_name)
            img = plt.imread(img_path)
            plt.imshow(img, cmap='gray')
            plt.title(category)
            plt.show()

display_sample_images(train_dir)
```

2. Génération des Données d'Entraînement, de Validation et de Test

Objectif : Préparer les générateurs de données pour charger les images de manière efficace.

```
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

print("Data generators created successfully")
```

Résultat :

```
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations
, rebuild TensorFlow with the appropriate compiler flags.
2024-07-23 01:59:45.368048: W tensorflow/compiler/tf2tensorrt/utils/py_utils
.cc:38] TF-TRT Warning: Could not find TensorRT
Found 5216 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
Data generators created successfully
Data loaded successfully
```

3. Chargement des Données pour les Modèles de Base

Objectif : Convertir les images en matrices et préparer les étiquettes pour les modèles de base.

```
tabnine: test | explain | document | ask
def load_data(folder):
    data = []
    labels = []
    categories = ['NORMAL', 'PNEUMONIA']
    for category in categories:
        path = os.path.join(folder, category)
        class_num = categories.index(category)
        for img_name in os.listdir(path):
            try:
                img_path = os.path.join(path, img_name)
                img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
                img = cv2.resize(img, (150, 150))
                data.append(img)
                labels.append(class_num)
            except Exception as e:
                print(e)
    data = np.array(data).reshape(-1, 150 * 150)
    labels = np.array(labels)
    return data, labels

train_data, train_labels = load_data(train_dir)
val_data, val_labels = load_data(val_dir)

print("Data loaded successfully")
```

Résultat : Data loaded successfully (visible sur l'image précédente)

4. Modèle SVM de Base

Objectif : Entraîner un modèle SVM sur les données de base et évaluer sa performance.

```
svm = SVC(kernel='linear')
svm.fit(train_data, train_labels)
svm_preds = svm.predict(val_data)

print(f'Accuracy du SVM: {accuracy_score(val_labels, svm_preds):.2f}')

print("SVM model trained and evaluated successfully")
```

5. Création et Entraînement d'un Modèle CNN Simple

Objectif : Créer et entraîner un modèle CNN pour la classification d'images.

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accu

print("CNN model created successfully")

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // val_generator.batch_size,
    epochs=10
)

print("CNN model trained successfully")
```

6. Validation Croisée pour le SVM

Objectif : Utiliser la validation croisée pour évaluer le modèle SVM.

```
cross_val_scores = cross_val_score(SVC(kernel='linear'), train_data, trair
print(f'Scores de validation croisée: {cross_val_scores}')
print(f'Moyenne des scores de validation croisée: {np.mean(cross_val_score
```

7. Optimisation des Paramètres avec Grid Search pour le SVM

Objectif : Trouver les meilleurs paramètres pour le modèle SVM en utilisant Grid Search.

```
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=2)
grid.fit(train_data, train_labels)

print(f'Best Parameters: {grid.best_params_}')
print(f'Best Score: {grid.best_score_:.2f}')
```

8. Réduction de Dimensionnalité avec PCA

Objectif : Réduire la dimensionnalité des données d'entrée en utilisant PCA et entraîner un modèle SVM.

```
pca = PCA(n_components=50)
train_data_pca = pca.fit_transform(train_data)
val_data_pca = pca.transform(val_data)

svm_pca = SVC(kernel='linear')
svm_pca.fit(train_data_pca, train_labels)
svm_pca_preds = svm_pca.predict(val_data_pca)

print(f'Accuracy du SVM avec PCA: {accuracy_score(val_labels, svm_pca_prec
```

9. Évaluation du Modèle CNN avec ROC-AUC

Objectif : Calculer le score ROC-AUC pour le modèle CNN et afficher la courbe ROC.

```
val_preds = model.predict(val_generator)
roc_auc = roc_auc_score(val_generator.classes, val_preds)
print(f'ROC-AUC Score pour le CNN: {roc_auc:.2f}')

fpr, tpr, _ = roc_curve(val_generator.classes, val_preds)
plt.plot(fpr, tpr, marker='.', label='CNN')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()

model.save('cnn_pneumonia_model.h5')
```

Explication des Modèles et Méthodes

Modèles Utilisés

1. **Support Vector Machine (SVM)** :
    - **Principe :** Un algorithme d'apprentissage supervisé utilisé pour la classification et la régression. Il trouve l'hyperplan optimal qui sépare les classes dans un espace de caractéristiques à haute dimension.
    - **Usage dans le projet :** Entraîné avec un noyau linéaire pour classer les images de radiographies en NORMAL et PNEUMONIA.

2. **Convolutional Neural Network (CNN)** :
    - **Principe :** Un type de réseau de neurones artificiels efficace pour le traitement des images. Les couches de convolution et de pooling permettent d'extraire et de réduire les caractéristiques pertinentes des images.
    - **Usage dans le projet :** Créé et entraîné pour la classification d'images de radiographies pulmonaires. Il comprend des couches de convolution, de pooling, une couche d'aplatissement, et des couches entièrement connectées.
3. **Principal Component Analysis (PCA)** :
    - **Principe :** Une méthode de réduction de la dimensionnalité qui transforme les données d'entrée en un ensemble de variables non corrélées appelées composantes principales, qui elles retiennent la plupart de la variance des données.
    - **Usage dans le projet :** Utilisé pour réduire la dimensionnalité des données avant d'entraîner un modèle SVM, ce qui aurait pu améliorer les performances et réduire le temps de calcul.

Méthodes d'Évaluation et d'Optimisation

1. **Validation Croisée (Cross-Validation)** :
    - **Principe :** Technique pour évaluer la performance d'un modèle d'apprentissage automatique en divisant les données en plusieurs sous-ensembles. Le modèle est entraîné sur une partie des données et testé sur une autre, de manière répetitive. La performance moyenne sur tous les sous-ensembles est calculée.
    - **Usage dans le projet :** Utilisée pour évaluer la performance du modèle SVM. Une validation croisée en 5 plis (k-fold) a été utilisée, ce qui signifie que les données ont été divisées en 5 parties, chaque partie étant utilisée comme ensemble de test une fois, et comme ensemble d'entraînement 4 fois.
2. **Recherche par Grille (Grid Search)** :
    - **Principe :** Une méthode d'optimisation hyperparamétrique qui explore systématiquement une grille d'hyperparamètres possibles pour trouver la combinaison optimale. Chaque combinaison est évaluée en utilisant une technique d'évaluation comme la validation croisée.
    - **Usage dans le projet :** Utilisée pour trouver les meilleurs hyperparamètres pour le modèle SVM, tels que le paramètre de régularisation C et le type de noyau (linear ou rbf).

Conclusion

En résumé, les tests ont été effectués pour visualiser les données, préparer les générateurs de données, charger les données dans un format utilisable pour les modèles, entraîner et évaluer un modèle SVM de base, créer et entraîner un modèle CNN, utiliser la validation croisée et la recherche de grille pour le SVM, réduire la dimensionnalité des données avec PCA, évaluer le modèle CNN avec ROC-AUC, et sauvegarder le modèle entraîné.

La validation croisée aurait assuré une évaluation robuste de la performance, tandis que le Grid Search aurait affiné les hyperparamètres pour obtenir le meilleur modèle possible.

Meilleure Approche Prévue :

Sans l'exécution complète, la combinaison du modèle CNN avec une évaluation ROC-AUC aurait pu fournir une meilleure compréhension des performances du modèle pour les raisons suivantes :

- **Mesure de Performance Agrégée :** L'ROC-AUC aurait permis d'évaluer comment le modèle CNN performe globalement sur tous les seuils de décision, donnant une mesure robuste de la qualité de la classification.

- **Évaluation des Trade-offs :** En visualisant la courbe ROC, nous aurions pu observer les trade-offs entre la sensibilité et la spécificité, aidant à ajuster le seuil de décision optimal en fonction des besoins cliniques (par exemple, préférer une sensibilité plus élevée pour ne pas manquer de cas de pneumonie).

- **Diagnostic des Modèles :** Si plusieurs modèles sont en compétition (par exemple, SVM vs CNN), comparer leurs AUC aurait aidé à sélectionner celui qui a la meilleure capacité discriminative.

Si l'exécution complète était possible, les résultats de la validation croisée et du Grid Search pour le SVM, ainsi que les performances du CNN évaluées par le score ROC-AUC, auraient été des indicateurs clés pour déterminer le meilleur modèle pour cette tâche.