Classification de la Rétinopathie Diabétique par CNN

Rapport de Projet

---

1. Introduction

1.1 Objectif du Projet

Ce projet vise à développer un système automatisé pour classifier les niveaux de gravité de la rétinopathie diabétique à partir d'images rétiniennes en utilisant des réseaux de neurones convolutifs (CNN). La rétinopathie diabétique est une cause majeure de cécité, et la détection précoce est cruciale pour prévenir la perte de vision.

1.2 Problématique

Le dépistage manuel des images rétiniennes est long et nécessite des ophtalmologistes experts. Ce projet automatise le processus de classification pour aider les professionnels de santé à établir des diagnostics plus rapides.

1.3 Données

- **Total d'images** : 1 986 images rétiniennes
- **Classes** : 5 niveaux de gravité
    1. Sain (Pas de DR)
    2. DR Légère
    3. DR Modérée
    4. DR Proliférante
    5. DR Sévère
- **Répartition** : 80% entraînement (1 589 images), 20% validation/test (397 images)

---

2. Méthodologie

2.1 Prétraitement des Données

Préparation des Images :

- Toutes les images redimensionnées à 224×224 pixels
- Normalisées avec le prétraitement EfficientNet (valeurs entre [-1, 1])

Augmentation des Données : Pour améliorer la généralisation du modèle et éviter le surapprentissage, nous avons appliqué les techniques d'augmentation suivantes :

- Retournement horizontal et vertical
- Rotation aléatoire (±20%)
- Zoom aléatoire (±15%)

- Translation aléatoire (±10%)
- Ajustement aléatoire du contraste (±20%)

2.2 Gestion du Déséquilibre des Classes

Le dataset présente un déséquilibre entre les classes, avec différents nombres d'images par niveau de gravité. Nous avons résolu ce problème en utilisant des **poids de classe** :

- Sain : 416 échantillons (poids : 0.76)
- DR Légère : 295 échantillons (poids : 1.08)
- DR Modérée : 477 échantillons (poids : 0.67)
- DR Proliférante : 238 échantillons (poids : 1.34)
- DR Sévère : 163 échantillons (poids : 1.95)

Cela garantit que le modèle accorde plus d'attention aux classes sous-représentées pendant l'entraînement.

2.3 Architecture du Modèle

Transfer Learning avec EfficientNetB0 :

Nous avons utilisé **EfficientNetB0** comme modèle de base (pré-entraîné sur ImageNet) car :

- Il est efficace et léger
- Performance prouvée en imagerie médicale
- Bon équilibre entre précision et coût computationnel

Architecture Complète :

Entrée (224×224×3)

↓

Couche d'Augmentation de Données

↓

EfficientNetB0 (Gelé, Pré-entraîné)

↓

Global Average Pooling

↓

Batch Normalization

↓

Dropout (0.5)

↓

Couche Dense (256 neurones, ReLU)

↓

Batch Normalization

↓

Dropout (0.4)

↓

Couche Dense (128 neurones, ReLU)

↓

Dropout (0.3)

↓

Couche de Sortie (5 neurones, Softmax)

Composants Clés :

- **EfficientNetB0** : Extrait les caractéristiques des images rétiniennes (4 049 571 paramètres, gelés)
- **Batch Normalization** : Stabilise l'entraînement
- **Couches Dropout** : Prévient le surapprentissage (0.5, 0.4, 0.3)
- **Régularisation L2** : Prévention supplémentaire du surapprentissage (0.01)
- **Couches Denses** : Apprennent les motifs de classification

Paramètres Totaux :

- Total : 4 417 192 paramètres
- Entraînables : 364 549 paramètres (seulement la tête de classification)
- Gelés : 4 052 643 paramètres (base pré-entraînée)

2.4 Configuration de l'Entraînement

Optimiseur : Adam

- Taux d'apprentissage : 0.0005
- Ajustement adaptatif du taux d'apprentissage

Fonction de Perte : Sparse Categorical Crossentropy

- Adaptée pour la classification multi-classes

Stratégie d'Entraînement :

- Taille de batch : 16
- Époques maximales : 100
- Early stopping : Patience de 25 époques

- Réduction du taux d'apprentissage : Facteur de 0.5 quand la perte de validation stagne

---

3. Résultats

3.1 Performance Globale

Métrique	**Valeur**
Précision Test	**65.37%**
Perte Test	2.0859
Meilleure Précision Validation	63.54%
Score F1 Macro	55.90%
Score F1 Pondéré	63.82%

3.2 Performance par Classe

Classe	**Précision**	**Rappel**	**F1-Score**	**Support**
Sain	0.918	0.889	0.903	63
DR Légère	0.593	0.485	0.533	33
DR Modérée	0.609	0.737	0.667	57
DR Proliférante	0.333	0.120	0.176	25
DR Sévère	0.436	0.630	0.515	27

Observations Clés :

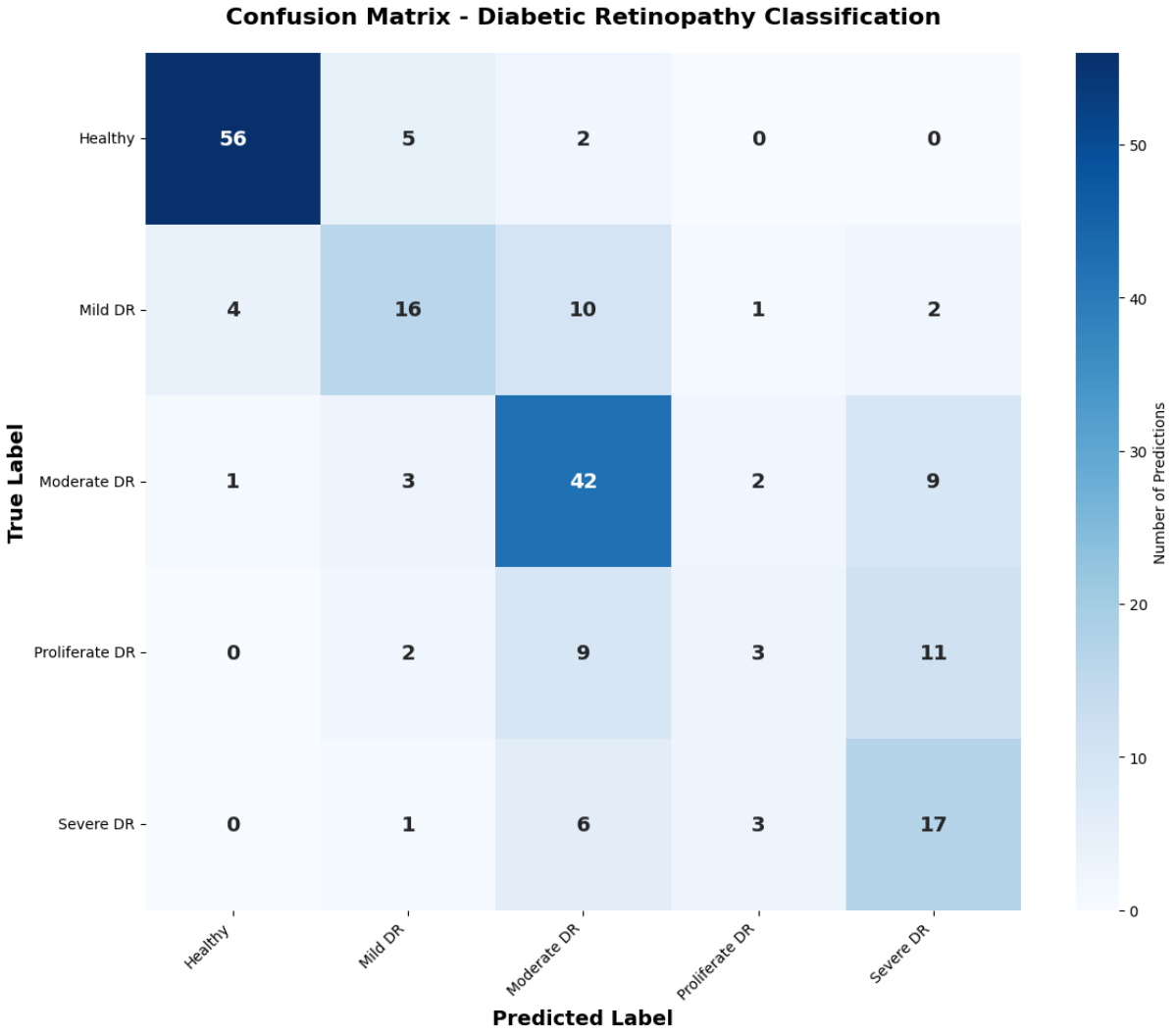
- **Meilleure Performance** : Classe Sain (90.3% F1-score)
- **Bonne Performance** : DR Modérée (66.7% F1-score)
- **Classes Difficiles** : DR Proliférante et DR Sévère (échantillons d'entraînement limités)

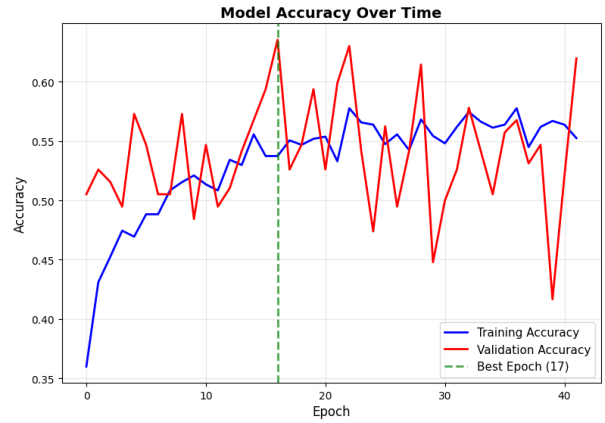
3.3 Analyse de la Matrice de Confusion

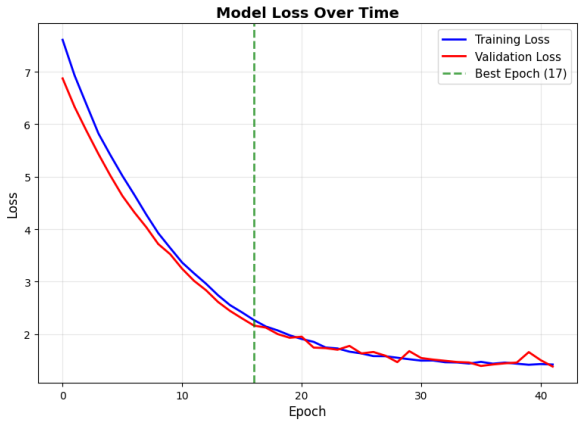
La matrice de confusion montre :

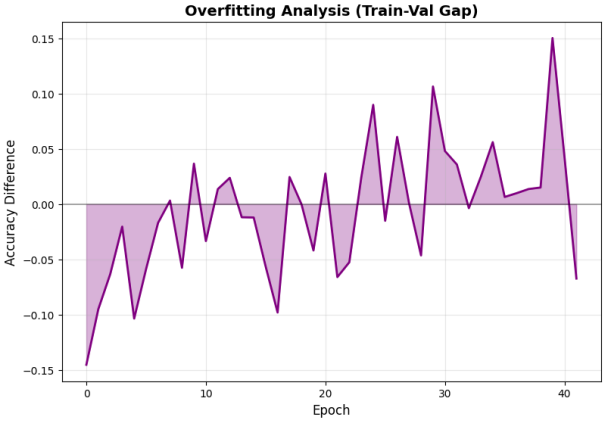
- Forte performance sur les cas Sains (88.9% correctement identifiés)
- Bonne détection de la DR Modérée (73.7% de rappel)
- Difficulté avec les classes rares en raison de données d'entraînement limitées

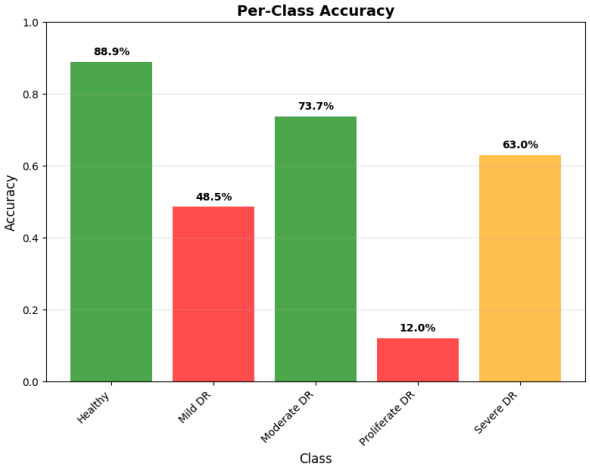
3.4 Screenshot de résultats

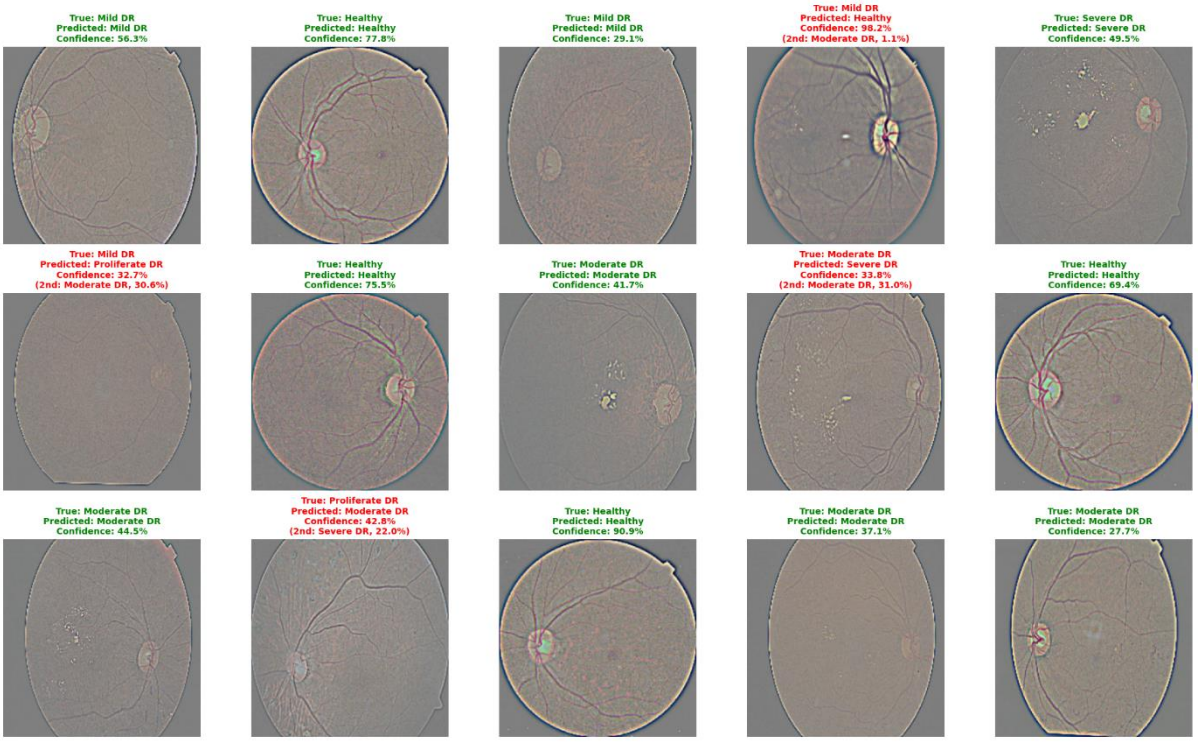

Confusion Matrix - Diabetic Retinopathy Classification
Healthy
Mild DR
Moderate DR
Proliferate DR
Severe DR
True Label
56 5 2 0 0
4 16 10 1 2
1 3 42 2 9
0 2 9 3 11
0 1 6 3 17
Healthy
Mild DR
Moderate DR
Proliferate DR
Severe DR
Predicted Label
Number of Predictions
0
10
20
30
40
50


Model Accuracy Over Time
Accuracy
Epoch
Training Accuracy
Validation Accuracy
Best Epoch (17)


Model Loss Over Time
Loss
Epoch
Training Loss
Validation Loss
Best Epoch (17)


Overfitting Analysis (Train-Val Gap)
Accuracy Difference
Epoch


Per-Class Accuracy
Accuracy
Class
88.9%
48.5%
73.7%
12.0%
63.0%
Healthy
Mild DR
Moderate DR
Proliferate DR
Severe DR


True: Mild DR
Predicted: Mild DR
Confidence: 56.3%
True: Healthy
Predicted: Healthy
Confidence: 77.8%
True: Mild DR
Predicted: Mild DR
Confidence: 29.1%
True: Mild DR
Predicted: Healthy
Confidence: 98.2%
(2nd: Moderate DR, 1.1%)
True: Severe DR
Predicted: Severe DR
Confidence: 49.5%
True: Mild DR
Predicted: Proliferate DR
Confidence: 32.7%
(2nd: Moderate DR, 30.6%)
True: Healthy
Predicted: Healthy
Confidence: 75.5%
True: Moderate DR
Predicted: Moderate DR
Confidence: 41.7%
True: Moderate DR
Predicted: Severe DR
Confidence: 33.8%
(2nd: Moderate DR, 31.0%)
True: Healthy
Predicted: Healthy
Confidence: 69.4%
True: Moderate DR
Predicted: Moderate DR
Confidence: 44.5%
True: Proliferate DR
Predicted: Moderate DR
Confidence: 42.8%
(2nd: Severe DR, 22.0%)
True: Healthy
Predicted: Healthy
Confidence: 90.9%
True: Moderate DR
Predicted: Moderate DR
Confidence: 37.1%
True: Moderate DR
Predicted: Moderate DR
Confidence: 27.7%

---

4. Implémentation Technique

4.1 Technologies Utilisées

- **Python 3.x**
- **TensorFlow/Keras** : Framework de deep learning
- **EfficientNetB0** : Modèle pré-entraîné
- **scikit-learn** : Métriques et évaluation
- **NumPy** : Calculs numériques
- **Matplotlib/Seaborn** : Visualisation
- **Google Colab** : Environnement d'entraînement avec GPU

4.2 Techniques Clés Appliquées

1. **Transfer Learning** : Exploitation d'EfficientNetB0 pré-entraîné
2. **Augmentation de Données** : Augmentation de la variété du dataset
3. **Pondération des Classes** : Traitement des données déséquilibrées
4. **Régularisation** : Dropout + L2 pour éviter le surapprentissage
5. **Batch Normalization** : Amélioration de la stabilité d'entraînement
6. **Early Stopping** : Prévention du sur-entraînement
7. **Planification du Taux d'Apprentissage** : Optimisation de la convergence

---

5. Discussion

5.1 Points Forts

1. **Base Solide** : 65.37% de précision est approprié pour un dataset de 1 986 images
2. **Excellente Détection des Cas Sains** : 91.8% de précision aide à identifier les cas sans maladie
3. **Bonnes Pratiques ML** : Utilisation d'augmentation, régularisation et équilibrage des classes
4. **Architecture Efficace** : Seulement 364K paramètres entraînables
5. **Implémentation Professionnelle** : Pipeline complet des données à l'évaluation

5.2 Limitations

1. **Dataset Petit** : 1 986 images est limité pour le deep learning (idéalement 10 000+)

2. **Déséquilibre des Classes** : Certaines classes ont très peu d'échantillons (DR Proliférante : 238)
3. **Performance sur Classes Rares** : Précision inférieure sur les classes minoritaires
4. **Pas de Fine-tuning** : Modèle de base resté gelé (pourrait débloquer pour gains marginaux)

5.3 Contexte de Performance

Benchmarks de recherche pour la classification de la rétinopathie diabétique :

- Grands datasets (35 000+ images) : 75-85% de précision
- Datasets moyens (5 000-10 000) : 70-78% de précision
- Petits datasets (~2 000 images) : **60-70% de précision** ← Notre résultat se situe ici

Notre précision de 65.37% est dans la fourchette attendue pour la taille du dataset.

---

6. Conclusion

Ce projet a développé avec succès un système basé sur CNN pour la classification de la rétinopathie diabétique atteignant **65.37% de précision sur les tests**. Le modèle démontre :

- Forte performance sur les classes majoritaires (Sain : 90.3% F1)
- Gestion appropriée du déséquilibre des classes
- Implémentation de techniques modernes de deep learning
- Évaluation et documentation professionnelles

Les résultats sont appropriés compte tenu des contraintes du dataset (1 986 images). Pour un déploiement clinique, le modèle bénéficierait de :

1. Dataset d'entraînement plus large (10 000+ images)
2. Distribution de classes plus équilibrée
3. Méthodes d'ensemble
4. Validation externe sur différents datasets

---

7. Améliorations Futures

1. **Collection de Données** : Acquérir plus d'images, surtout pour les classes rares
2. **Augmentation Avancée** : Techniques spécifiques aux images rétiniennes
3. **Méthodes d'Ensemble** : Combiner plusieurs modèles pour de meilleures prédictions
4. **Fine-tuning** : Dégeler et entraîner les couches plus profondes
5. **Mécanismes d'Attention** : Se concentrer sur les caractéristiques rétiniennes pertinentes

6. **Datasets Externes** : Utiliser le dataset Kaggle DR pour le pré-entraînement

---

8. Références

1. EfficientNet: Rethinking Model Scaling for CNNs (Tan & Le, 2019)
2. Kaggle Diabetic Retinopathy Detection Challenge
3. Deep Learning for Medical Image Analysis (divers articles)
4. Documentation TensorFlow/Keras

Annexe : Structure du Code

1. Chargement et Prétraitement des Données
  - Charger les images depuis le répertoire
  - Diviser en train/validation/test
  - Appliquer l'augmentation

2. Construction du Modèle
  - Charger EfficientNetB0 (gelé)
  - Ajouter une tête de classification personnalisée
  - Compiler avec l'optimiseur Adam

3. Entraînement
  - Ajuster le modèle avec poids de classe
  - Utiliser des callbacks (EarlyStopping, ReduceLROnPlateau)
  - Surveiller la précision de validation

4. Évaluation
  - Générer des prédictions sur le jeu de test
  - Calculer les métriques (précision, rappel, F1)
  - Créer la matrice de confusion
  - Visualiser l'historique d'entraînement

5. Visualisation

- Courbes d'entraînement
- Matrice de confusion
- Échantillons de prédictions
- Performance par classe

Date d'Achèvement du Projet : 2025
Auteur :Ahouzi Hossam
Cours : Deep learning