

CLASSIFICATION MULTI-CLASSES PAR SVM ET PCA

Le projet repose sur le "Pokémon Classification Dataset" de Kaggle

- Nombre total d'images : 6820
- Nombre de classes : 150

Assitan



SOMMAIRE DE LA PRESENTATION

01

Objectifs & Contexte

Défi de la classification de 150 classes
Justification du choix

02

Méthodologie & Préparation

Prétraitement de l'image 64 * 64 et
étape critique de PCA (de 4096 à 150
dimensions).

03

Modélisation & Optimisation

AI subset that learns from data to make
predictions.

04

Résultats & Perspectives

Utilisation du SVC Kernel RBF et
optimisation des hyperparamètres par
GridSearchCV

OBJECTIFS STRATÉGIQUES

ET JUSTIFICATION DU MODÈLE

Notre projet s'articule autour de deux objectifs majeurs et un choix stratégique du modèle.

01

Objectif Technique (Maîtrise du Pipeline)

Le premier objectif était technique: maîtriser la chaîne complète de traitement d'images sans Deep Learning, de la préparation à l'évaluation.

04

Objectif Applicatif (Le Défi)

Le défi applicatif était de construire un classifieur multi-classes capable de distinguer les 150 classes différentes de Pokémon de la première génération

JUSTIFICATION DU CHOIX DU SVM

Le SVM a été sélectionné pour sa robustesse historique et son principe d'efficacité théorique : la maximisation de la marge. Il est aussi naturellement bien adapté à la gestion de la haute dimensionnalité des données d'image

Cependant, nous savions que son succès dépendrait entièrement de la réduction de dimensionnalité par la PCA.

DATASET, PRÉTRAITEMENT, ET DIMENSION INITIALE

Passons à l'étape la plus coûteuse en ressources : la préparation des données.

Le projet repose sur le "Pokémon Classification Dataset".

Ce ratio de seulement 45 images par classe est le principal défi, car il prédit des difficultés de généralisation.

Nous avons standardisé la taille à 64x64, un compromis pour réduire la charge de calcul.

L'aplatissement (flattening) nous donne un vecteur initial de 4096 dimensions. C'est ce problème que nous avons dû résoudre avec l'étape suivante : la PCA.

```
# 3) TRANSFORMATIONS PYTORCH

transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.Grayscale(num_output_channels=1),
    transforms.ToTensor()
])

# Chargement du dataset
dataset = datasets.ImageFolder("/content/pokemonclassification/PokemonData", transform=transform)

print("Taille dataset :", len(dataset))

# 4) SPLIT : TRAIN / VAL / TEST

train_size = int(0.7 * len(dataset))
val_size   = int(0.15 * len(dataset))
test_size  = len(dataset) - train_size - val_size
```

LEARN MORE

PIVOT TECHNIQUE RÉDUCTION PAR PCA

```
# 5) CONVERSION EN MATRICES NUMPY POUR SVM

def to_numpy(loader):
    X, y = [], []
    for imgs, labels in loader:
        imgs = imgs.view(imgs.size(0), -1) # flatten
        X.append(imgs.numpy())
        y.append(labels.numpy())
    return np.vstack(X), np.hstack(y)

print("Conversion en numpy...")

X_train, y_train = to_numpy(train_loader)
X_val, y_val = to_numpy(val_loader)
X_test, y_test = to_numpy(test_loader)

print("Dimensions X_train :", X_train.shape)
```

Face aux 4096 dimensions, la PCA est devenue le pivot de notre méthodologie.

Justification Théorique :

Problème : L'espace 4096D cause la Malédiction de la Dimensionnalité (données éparses, entraînement prohibitif)

Rôle de la PCA : Réduction du bruit et décorrélation des variables.

Stratégie de Réduction :

Choix Stratégique :

N_components = 150 composantes principales

Impact : Réduction drastique de la dimension, de 4096--150 features. [Insérer la Capture 2 : Code PCA + X_train_pca.shape]

MODÉLISATION ET OPTIMISATION DU SVC

Notre modèle SVM est maintenant prêt à opérer dans cet espace compact.
L'étape suivante était son optimisation.

Modèle et Kernel RBF

Modèle : SVC(Support Vector Classifier)
Kernel : RBF (Radial Basis Function). Ce noyau permet la séparation non-linéaire des données

Optimisation par GridSearchCV

Méthode : Recherche par grille (CV=3) testant 36 combinaisons de C et Meilleurs paramètres :{'C': 50, {'gamma': 'scale'}.

INTERPRETATION

```
# 7) GRIDSEARCH POUR TROUVER LES MEILLEURS PARAMÈTRES SVM

params = {
    'C': [2, 5, 8, 12, 15],
    'gamma': ['scale', 'auto', 0.001, 0.0005]
}

print("Lancement GridSearch... ça peut prendre 2-5 min selon GPU...")

grid = GridSearchCV(
    SVC(kernel='rbf'),
    params,
    cv=3,
    n_jobs=-1
)

grid.fit(X_train_pca, y_train)

print("Best params :", grid.best_params_)
print("Best CV accuracy :", grid.best_score_)

best_svm = grid.best_estimator_
```

RÉSULTAT CLÉ 8

ANALYSE DES FAIBLESSES

✓ Résultat Clé : Accuracy Test

Passons à l'évaluation finale sur le jeu de test.

Accuracy Test Finale : 0.20 (20%).

Validation : Ce score est significativement > 0.67 (aléatoire) , prouvant l'extraction d'information visuelle.

Bilan : Une Accuracy de 21% est insuffisante pour une application pratique (erreurs dans $> 80\%$ des cas).

✓ Analyse Détailée des Faiblesses

Problème : Les 150 classes sont trop proches dans l'espace réduit(150D), ce qui empêche une séparation nette.

*Faible Recall : De nombreuses classes (ex: Abra, Charizard) affichent un Recall = (0.20) signifiant qu'elles n'ont jamais été correctement identifiées

```
plt.figure(figsize=(12, 10))
sns.heatmap(cm, cmap="Blues", annot=False)
plt.title("Matrice de confusion – Pokémon SVM Classifier")
plt.show()

print("Projet terminé ! 🎉")
```

	139	0.33	0.38	0.35	8
...	140	0.00	0.00	0.00	9
	141	0.30	0.30	0.30	10
	142	0.27	0.43	0.33	7
	143	0.10	0.14	0.12	7
	144	0.00	0.00	0.00	10
	145	0.50	0.33	0.40	3
	146	0.33	0.20	0.25	5
	147	0.50	0.67	0.57	6
	148	0.33	0.33	0.33	6
	149	0.00	0.00	0.00	3
	accuracy			0.20	1023
	macro avg	0.21	0.20	0.19	1023
	weighted avg	0.23	0.20	0.20	1023

BILAN CRITIQUE & PERSPECTIVES

Bilan Critique :

Limite Structurelle :

L'approche SVM+PCA est limitée face à la complexité des images et au grand nombre de classes

Échec du Feature Engineering Manuel :

La PCA a fourni des caractéristiques brutes qui ne capturent pas les détails spatiaux nécessaires à la distinction.

Nécessité : Seul l'apprentissage hiérarchique d'un CNN peut extraire des caractéristiques pertinentes (formes, textures) automatiquement.

Perspectives d'Amélioration:

Recommandation Majeure : Transition vers les Réseaux de Neurones Convolutifs (CNN) (ex: ResNet, VGG).

Alternative : Utiliser un CNN pré-entraîné comme Feature Extractor à la place de la PCA.



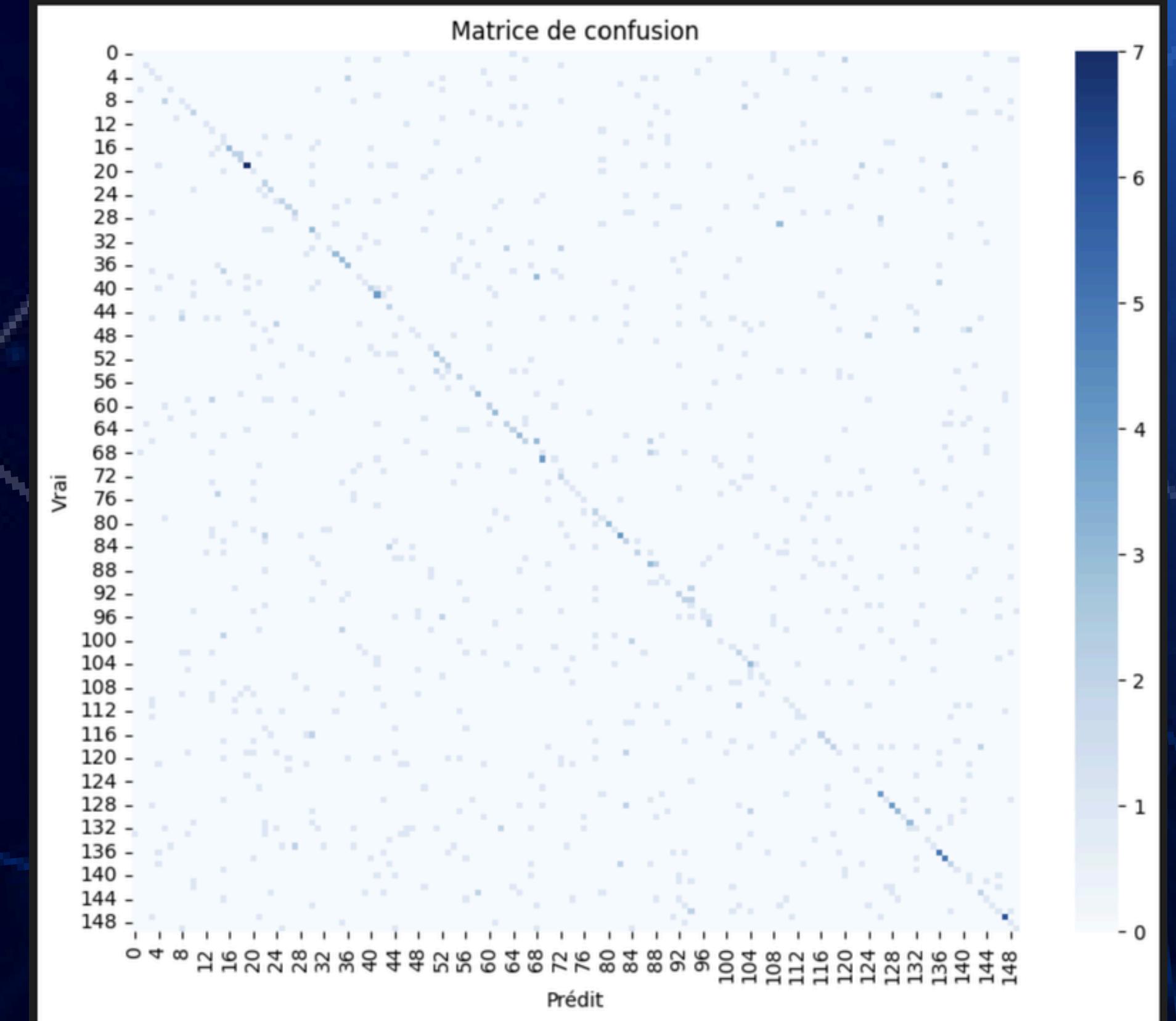
MATRICE DE CONFUSION

Matrice de Confusion

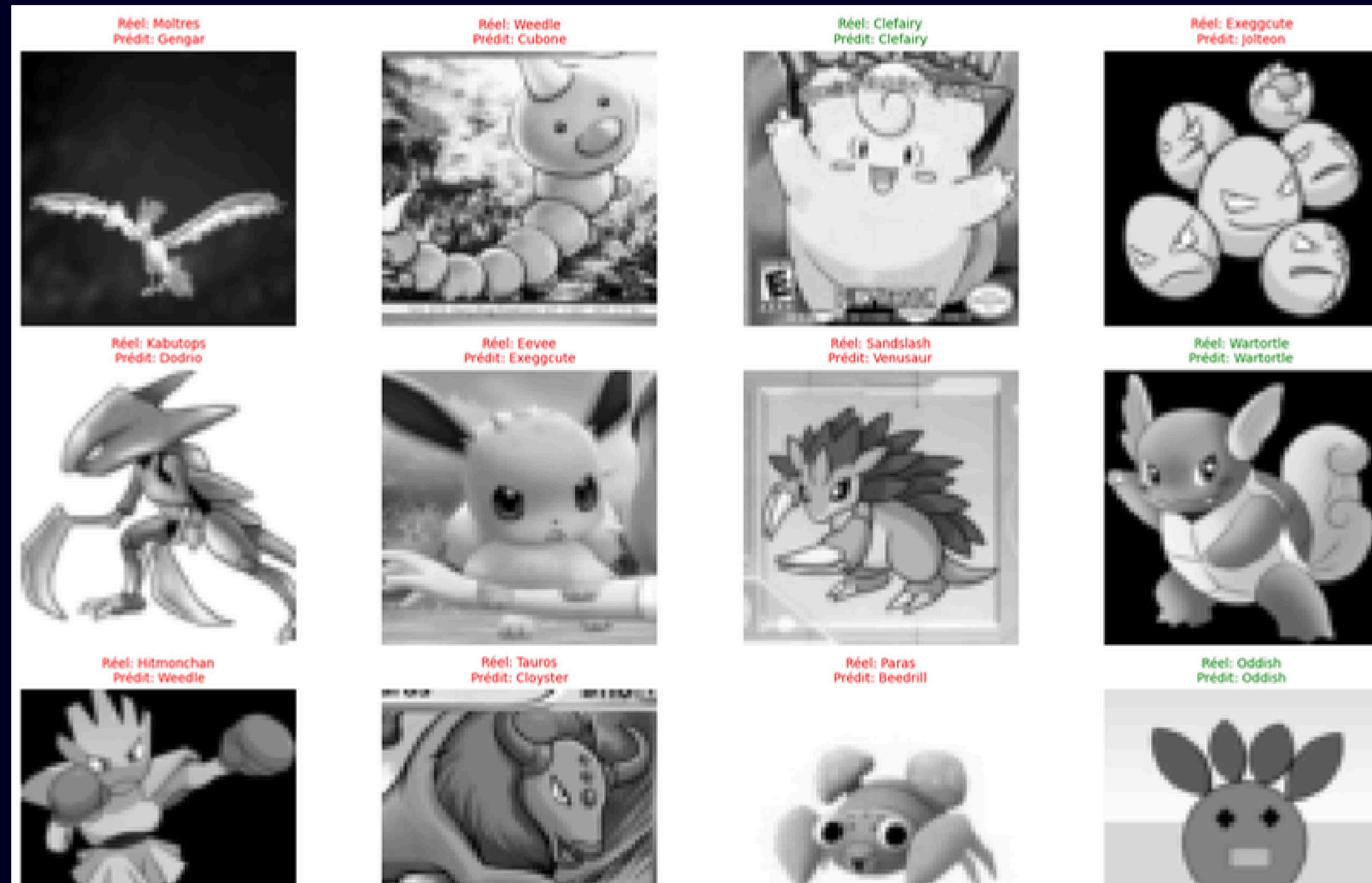
Observation : Très peu de concentration sur la diagonale (performance), confirmant la difficulté à séparer les 150 classes.

Cause : L'approche SVM+PCA n'est pas adaptée à l'extraction de caractéristiques discriminantes dans des problèmes visuels complexes.

C'est la preuve visuelle que l'approche SVM+PCA a atteint sa limite structurelle.



RESULTAT DU PREDICTION



CONCLUSION

Le pipeline technique a été exécuté avec succès et optimisé (GridSearchCV), mais l'approche SVM+PCA s'est avérée structurellement limitée, obtenant une performance modérée de 19.26%

Le projet démontre la nécessité de passer aux modèles basés sur les CNN pour les problèmes modernes de Vision par Ordinateur."Merci pour votre attention."Je vous remercie pour votre attention et je suis prêt à répondre à vos questions."



MERCI
POUR VOTRE ATTENTION
