# Rapport Projet Kaggle

Rémi Colin, Florian Grivet

ENSEEIHT & INSA Toulouse - 4ModIA
2022-2023

# Table des matières



# Table des figures

# 1 Différentes techniques utilisées

## 1.1 Dataset utilisé

Le dataset utilisé est composé d'une partie d'entrainement de 4000 images prises du dataset ImageNet avec leur label associé et d'une partie de test composée de 1082 images.
Il y a quatre types de label pour les images :
- 1 : Choux-fleur
- 2 : Tente de montagne
- 3 : Choux
- 4 : Rayon de miel

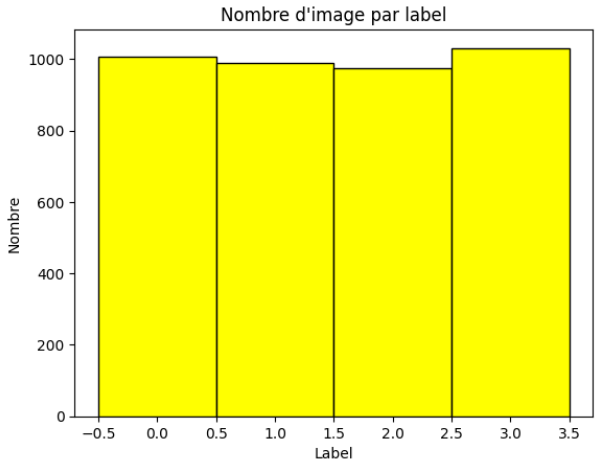


FIGURE 1 – Répartition des données d'entraînement

## 1.2 Augmentation de données

Le dataset étant composé de peu d'images, nous avons donc décidé de faire de l'augmentation de données. Cette technique consiste à générer de nouvelles données à partir des données d'entrainement en appliquant de multiples transformations à ces dernières afin d'étoffer et de diversifier la base d'entrainement initiale.

Pour ce faire, nous avons implémenté différentes transformations :
- transformation de base : renvoyant l'image initiale,
- random rotation : rotationnant l'image d'un angle aléatoire,
- random translation : translatant l'image aléatoirement selon les axes x et y,
- random zoom : zoomant l'image d'une échelle aléatoire,
- horizontal flip : renvoyant le miroir horizontal de l'image initiale,
- vertical flip : renvoyant le miroir vertical de l'image initiale,
- random brightness contrast : changeant la luminosité et le contraste de manière aléatoire,
- add random noise : ajoutant du bruit aléatoirement à l'image,
- convert to grayscale : convertissant en niveaux de gris l'image,
- random crop : découpant l'image d'une taille aléatoire.

Nous utilisons néanmoins uniquement quatre transformations lors de l'entrainement des données par manque d'espace mémoire. Nous avons trouvé que les quatres les plus pertinentes sont : transformation de base, random zoom, horizontal flip, vertical flip. Nous avons donc utilisé ces transfomations pour l'entrainement de chacun de nos modèles.

De plus, toutes les transformations redimensionnent nos images en (3x224x224) et les transforment en tenseur pytorch.

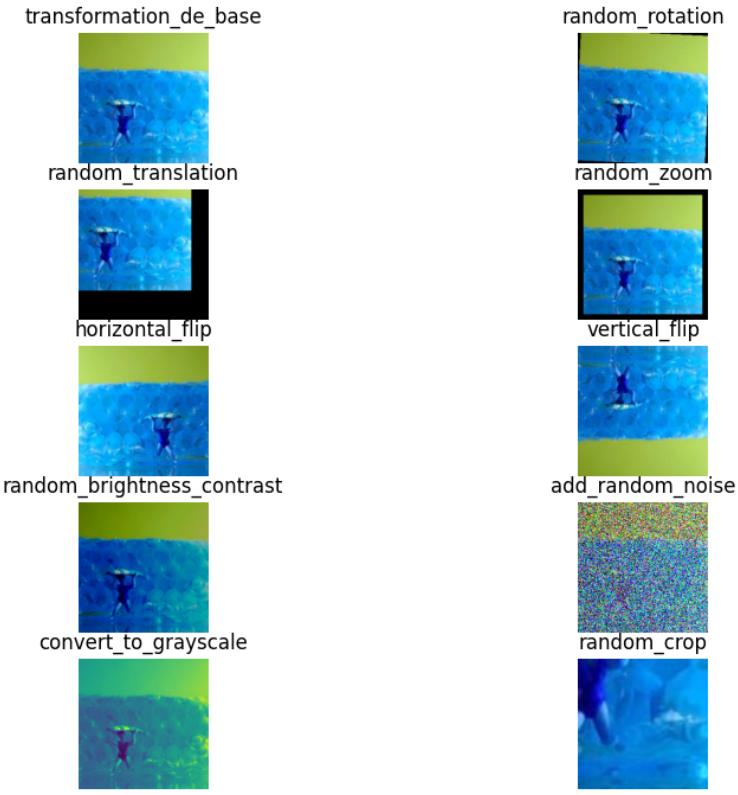


FIGURE 2 – Transformations implémentées

## 1.3 Modèles utilisés

Nous avons testés différentes architectures de réseaux de neurones : AlexNet, LeNet et ResNet avec différents paramètres dont les meilleurs seront présentés dans la partie suivante.
Cependant, le réseau de neurones ResNet donnant les meilleurs résultats, nous avons décidé de faire du transfert learning en utilisant le réseau de neurones ResNet50 afin de gagner du temps et de l'espace. Le transfert d'apprentissage est une technique de Machine Learning qui utilise les connaissances d'un modèle pré-entraîné sur une tâche pour améliorer l'apprentissage sur une autre tâche similaire. Cela permet de gagner du temps et d'améliorer les performances avec moins de données.

Puisque nous avions beaucoup de données d'entrainement grâce à l'augmentation de données, nous avons décidé de séparer nos données afin d'avoir un jeu de validation.

Nous avons également utilisé un système de vote pour améliorer les prédictions : on prédit chaque image de test avec nos cinq meilleurs modèles sur le jeu de validation et donnons le label le plus représenté à notre image.

### 1.3.1 AlexNet

Fonction de convolution :

| Couche | Nombre de canal en entrée | Taille d'un canal | Taille du kernel | Stride | Padding |
|---|---|---|---|---|---|
| Conv2D | 3 | 224x224 | 11x11 | 4 | 0 |
| MaxPool2D | 96 | 54x54 | 3x3 | 2 | 0 |
| Conv2D | 96 | 26x26 | 5x5 | 1 | 2 |
| MaxPool2D | 256 | 26x26 | 3x3 | 2 | 0 |
| Conv2D | 256 | 12x12 | 3x3 | 1 | 1 |
| Conv2D | 384 | 12x12 | 3x3 | 1 | 1 |
| Conv2D | 384 | 12x12 | 3x3 | 1 | 1 |
| MaxPool2D | 256 | 12x12 | 3x3 | 2 | 0 |

Fonction d'activation :

| Fonction | Nombre de sortie |
|---|---|
| Flatten | 256x5x5 = 6400 |
| Dense | 4096 |
| Dense | 4096 |
| Dense | 4 |

### 1.3.2 LeNet

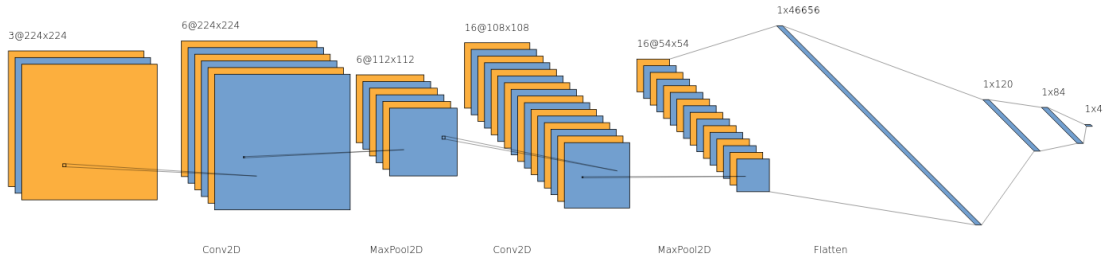


FIGURE 3 – Architecture LeNet

### 1.3.3 ResNet

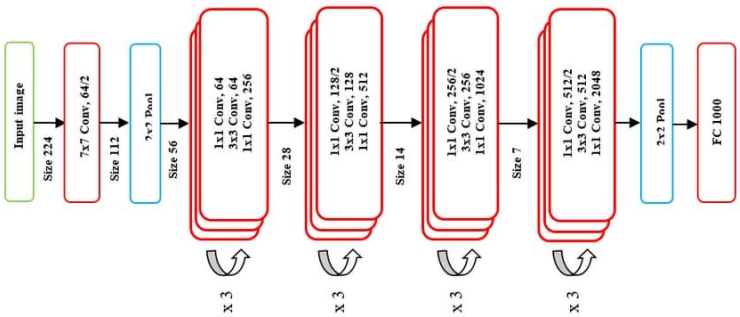


FIGURE 4 – Architecture ResNet50

## 2 Résultats obtenus

Pour chacun de nos résultats, nous avons utilisé un batch de 64, la CrossEntropyLoss comme fonction de perte car couramment utilisée dans des problèmes de classification comme le nôtre et le même optimizer : SGD avec un learning rate de $0,01$ et un momentum de 0.9.

Nous avons obtenu les meilleurs résultats avec :

| Modèle | Nombre d'epochs | Temps par epochs | Validation accuracy | Kaggle accuracy |
|---|---|---|---|---|
| AlexNet | 30 | 40s | $90,66\%$ | $86,69\%$ |
| LeNet | 30 | 20s | $72,17\%$ | $66,91\%$ |
| ResNet | 30 | 2min45s | $88,47\%$ | $88,17\%$ |
| ResNet Transfert Learning | 10 | 2min | $97,94\%$ | $97,23\%$ |

Les meilleurs résultats ont été trouvés avec le modèle de transfert learning, ce qui est normal car le modèle pré-entrainé possède des connaissances préalables et une meilleure généralisation en adaptant ses connaissances aux données d'entrainement.

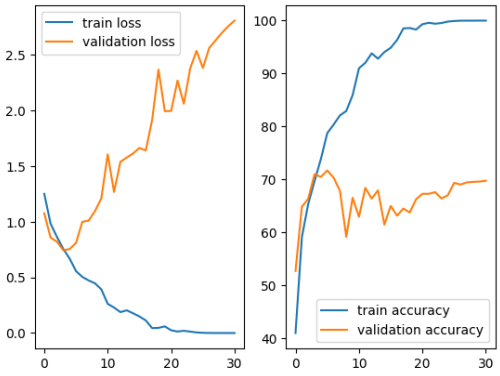


FIGURE 5 – LeNet

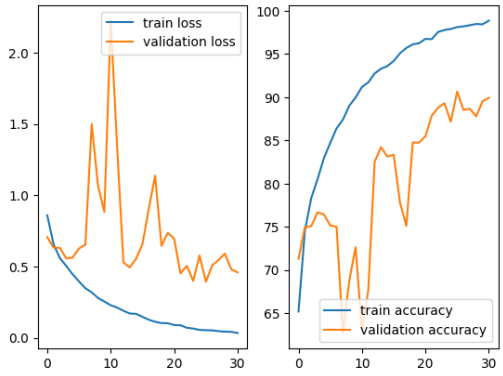


FIGURE 6 – AlexNet

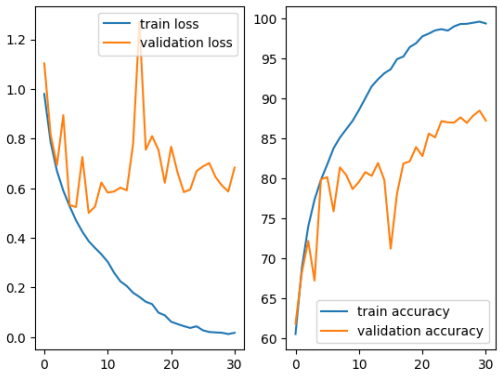


FIGURE 7 – ResNet

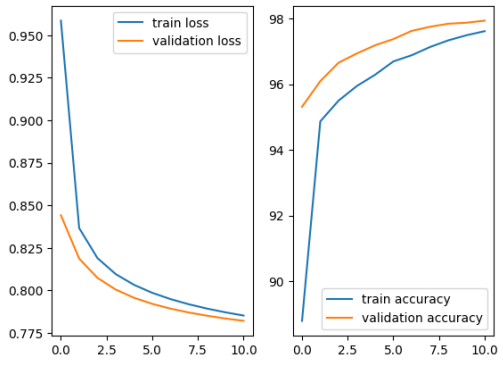


FIGURE 8 – ResNet avec Transfert Learning

# 3 Est-ce qu'on peut faire confiance à son modèle ?

Ces dernières années, avec l'avancée fulgurante de l'Intelligence Artificielle et leur application dans de multiples secteurs, garantir la robustesse des modèles est devenu une priorité, surtout lorsque des décisions critiques peuvent être prises. Garantir la robustesse des modèles implique la mise en place de procédure de contrôle pour détecter les valeurs atypiques ou les anomalies dans la base d'entrainement et de choisir judicieusement la fonction perte lors de l'entrainement du modèle. De plus, des mécanismes de détection d'anomalie sont primordiaux afin d'éviter de prendre des décisions biaisées ou basées sur des situations atypiques, non représentées dans la base d'apprentissage.

Pour commencer, il est crucial de détecter, dès la phase d'apprentissage, les valeurs atypiques ou anomalies dans la base de données utilisée afin d'évaluer la robustesse du modèle. Ces valeurs, non comprises dans la base d'entrainement, peuvent fauser la prédiction de l'algorithme et conduire à des résultats indésirables et imprévisibles. Traiter ces valeurs de manière appropriée est nécessaire et il est donc crucial de mettre en place des procédures de contrôle des données. Ces procédures peuvent permettre de corriger les erreurs, d'éliminer les données anormales ou de prendre en compte ou non certaines données existantes en fonction de leur conformité avec la base de données d'apprentissage.

Ensuite, le choix de la fonction perte est très important dans la robustesse du modèle créé. Cette dernière définit la manière dont les potentielles erreurs sont calculées/mesurées puis minimisées lors de l'apprentissage. Plus l'application du modèle est sensible, plus il est important de bien réfléchir et choisir correctement la fonction perte afin qu'elle pénalise de manière appropriée les erreurs importantes et/ou les anomalies dans la base d'entrainement. Ainsi, une fonction perte donnant plus de poids aux exemples moins fréquents ou mal placés serait robuste car permettrait d'éviter des décisions erronées dans des situations atypiques.

Enfin, des mécanismes de détection d'anomalies doivent également être intégrées afin d'assurer la robustesse d'un modèle. En effet, l'Intelligence Artificielle doit être en mesure de reconnaître des situations non conformes à sa base d'apprentissage lorsqu'elle est utilisée dans un contexte réel afin de ne pas proposer des décisions aléatoires, basées sur un scénario qui lui est inconnu et qui pourrait entraîner de graves conséquences. Différentes techniques de détections de situations atypiques peuvent être mises en place afin d'éviter ces potentielles erreurs de décision telles que l'apprentissage non supervisé ou l'analyse statistique.

Pour conclure, il est primordial de créer des modèles d'Intelligence Artificielle robustes, surtout dans des secteurs sensibles afin d'éviter de graves conséquences. La mise en place de procédures de contrôle ainsi qu'un choix judicieux de fonction de perte pour minimiser les erreurs importantes lors de l'apprentissage permet d'augmenter cette robustesse. De plus, installer un mécanisme de détection d'anomalie permet d'avertir l'utilisateur d'une potentielle erreur de décision et augmenter encore une fois la fiabilité du modèle.