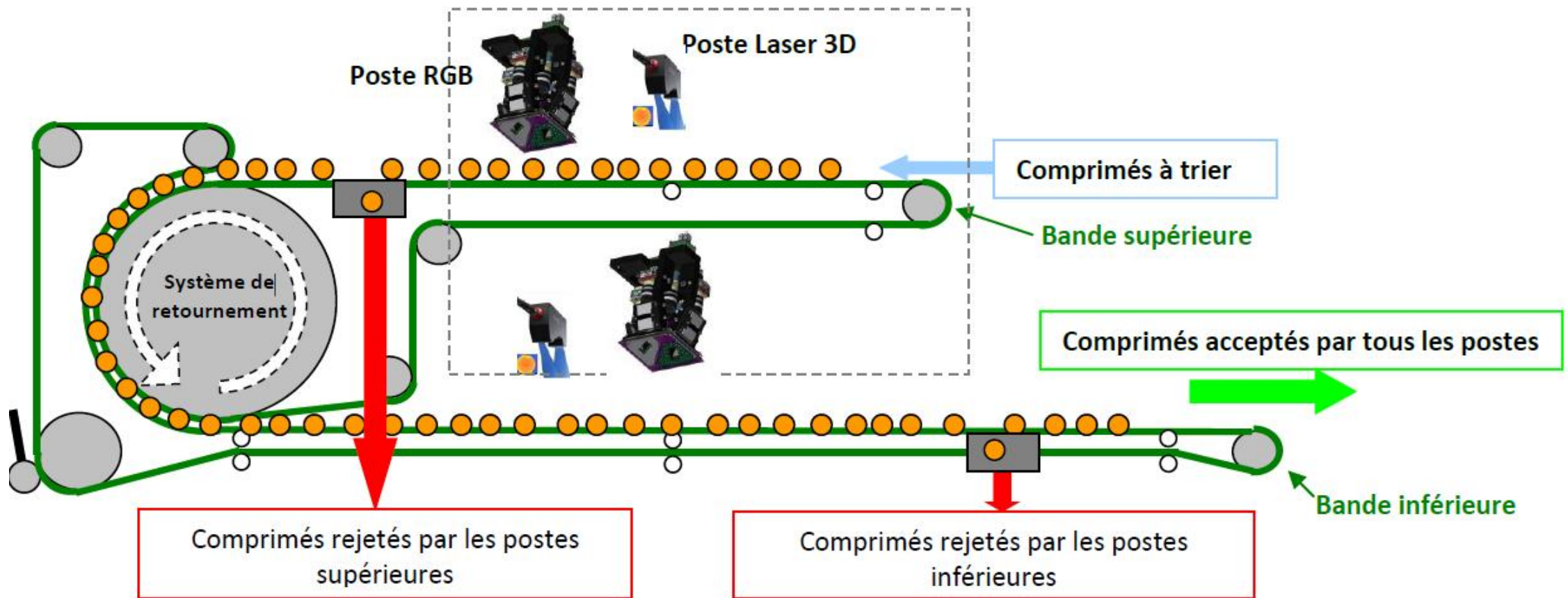


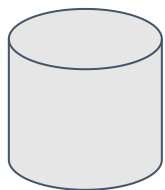
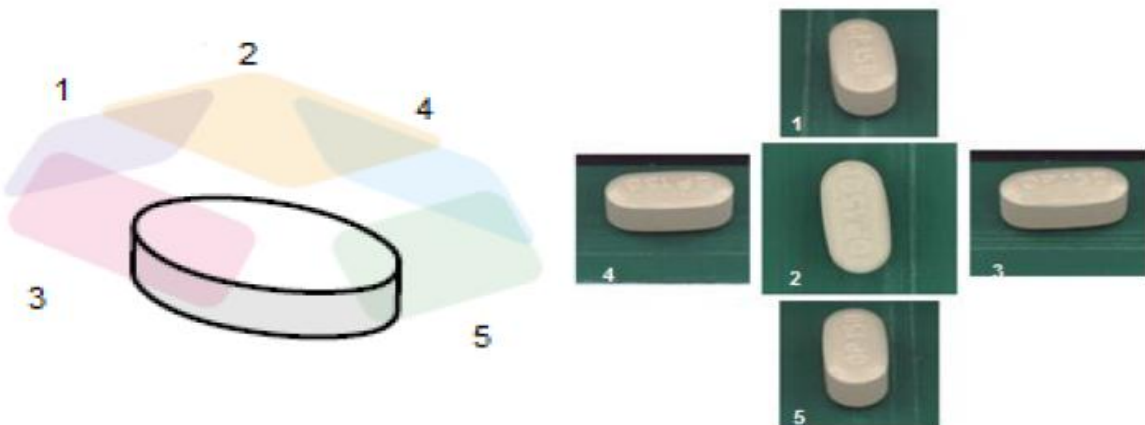
CAS D'UTILISATION PRODITEC-1



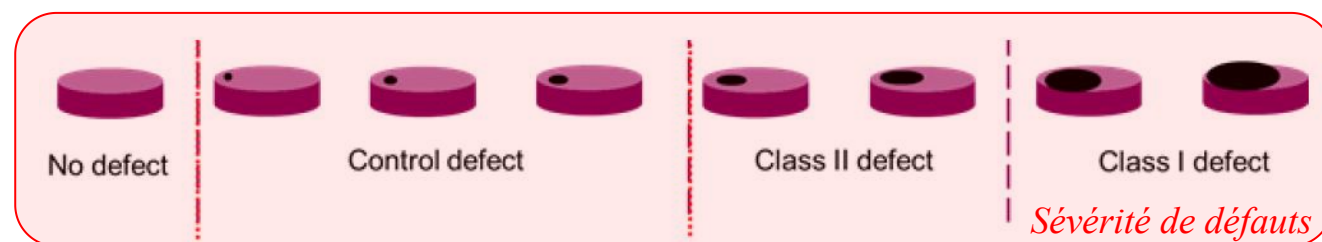
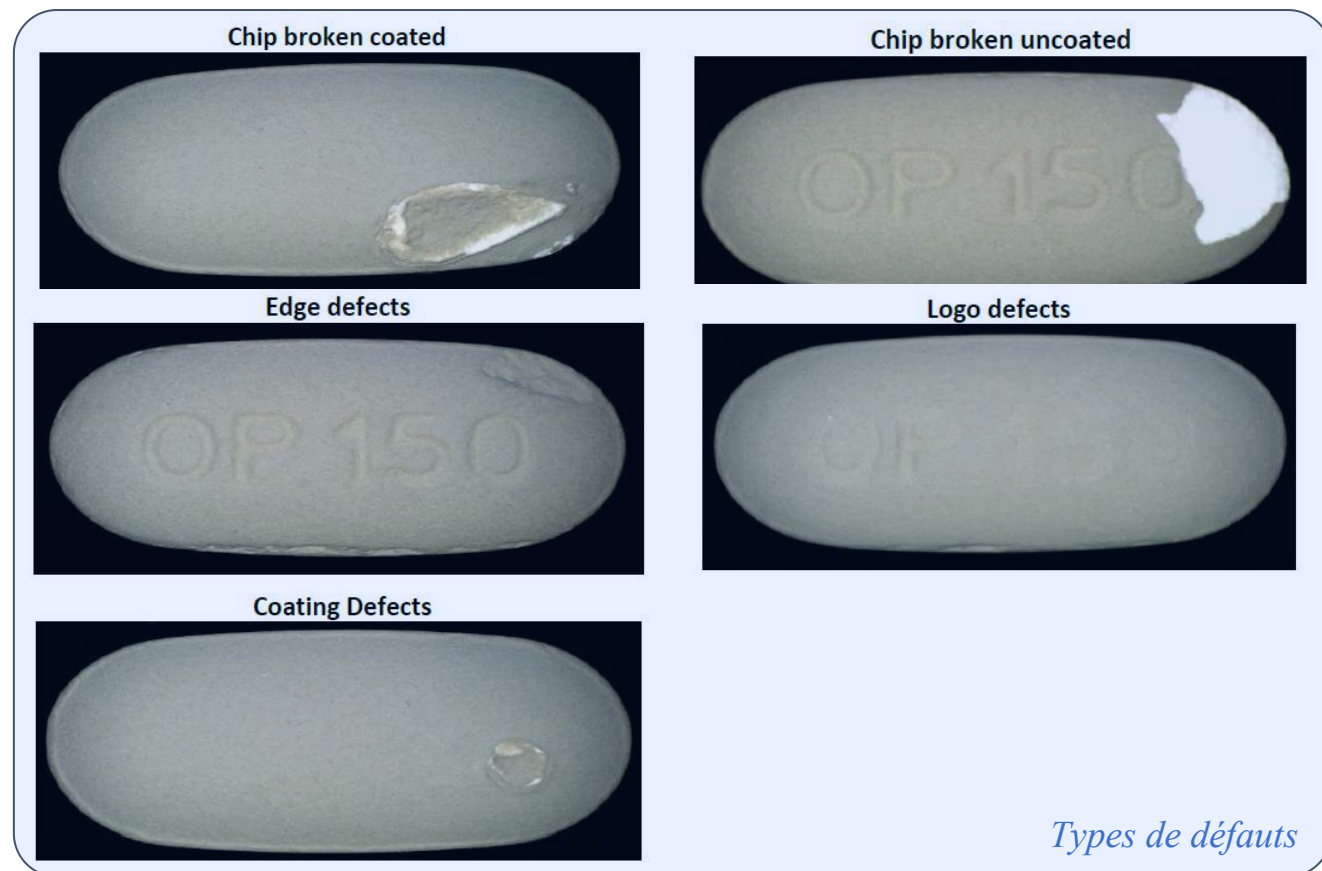
Description du cas d'utilisation



Description du cas d'utilisation



Training: 1002 images
Testing: 2744 images

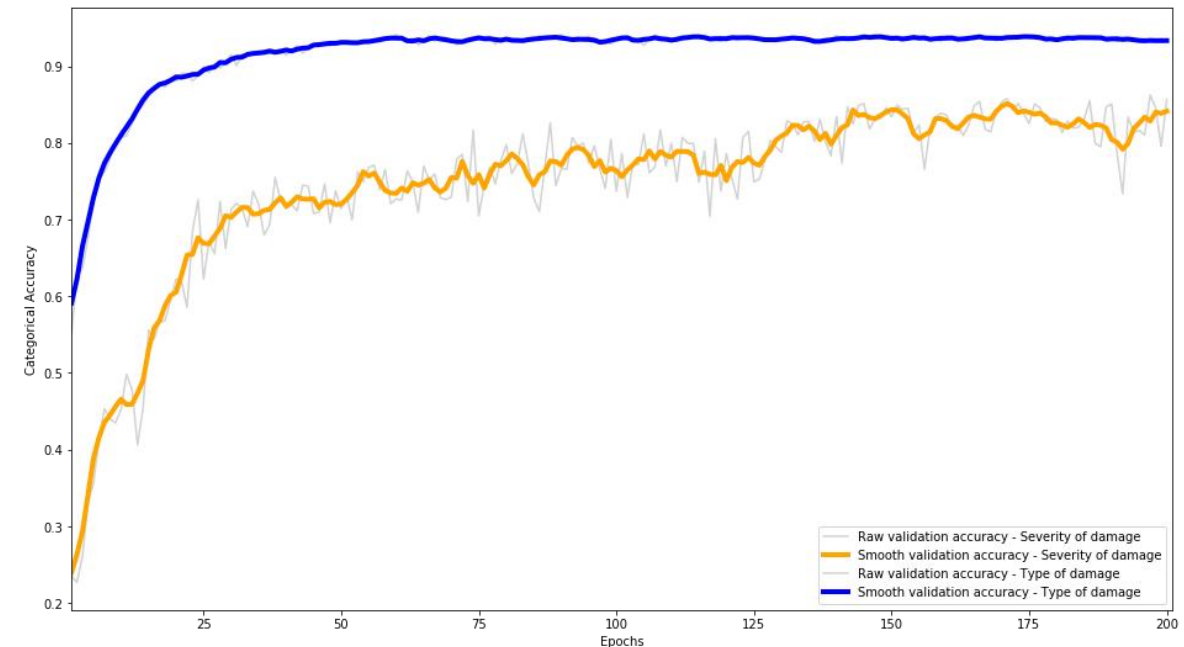
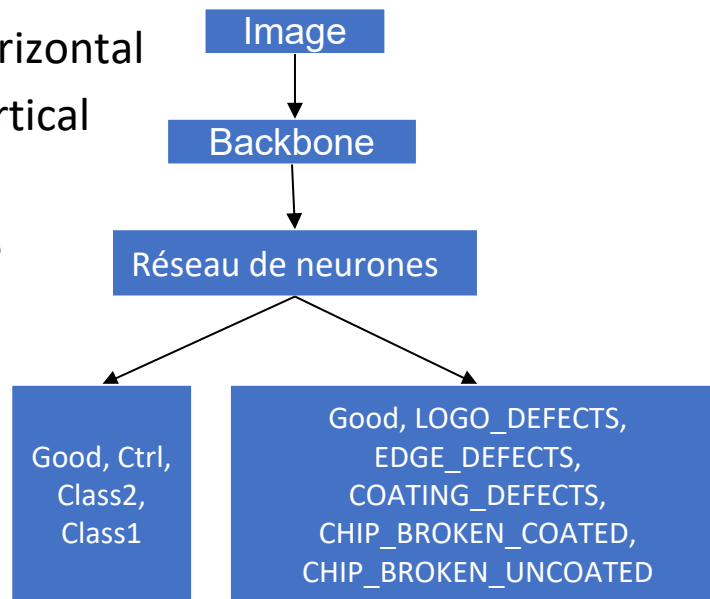


Approche 1 : Le modèle

Nous avons créé un modèle utilisant un backbone (modèle pré-entraîné) à 2 sorties, chacune subdivisée en respectivement 4 et 6 classes.

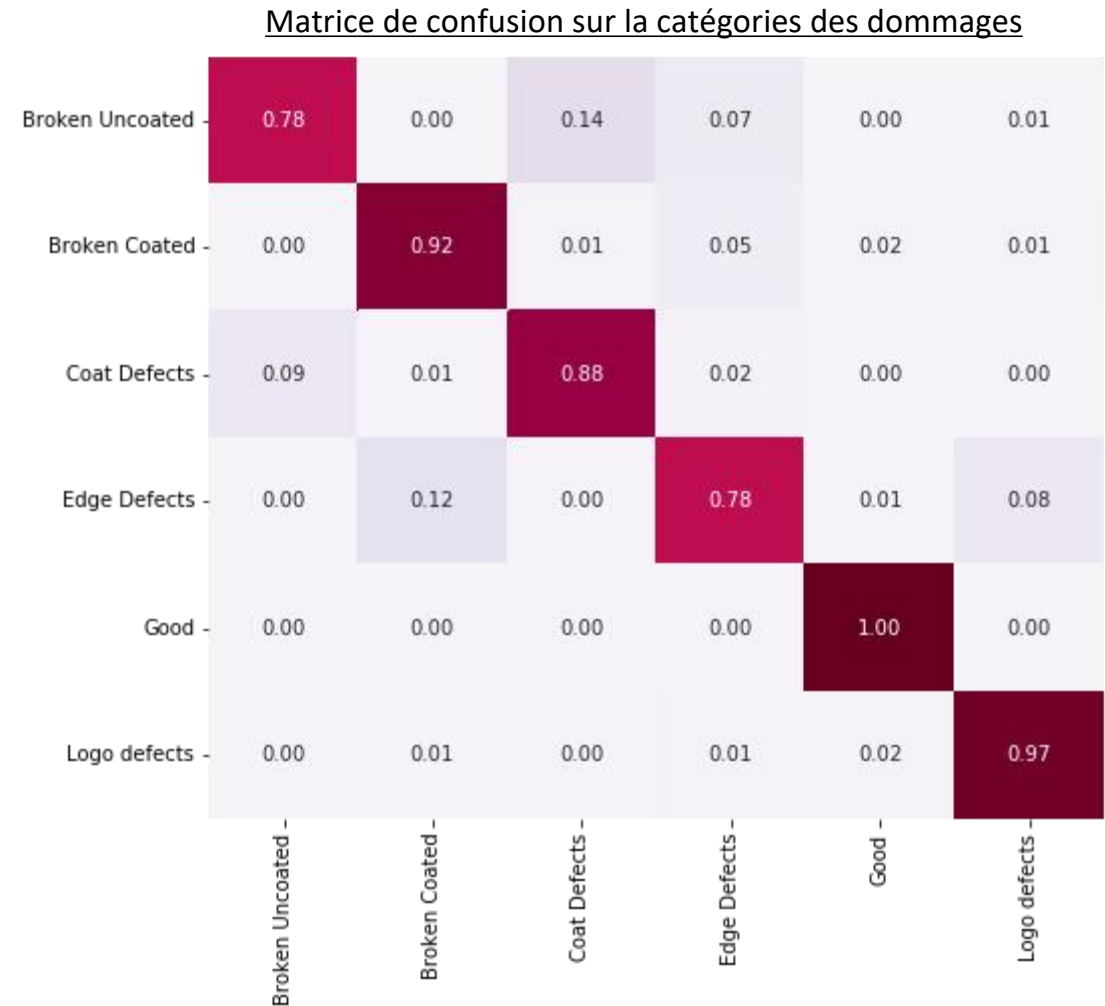
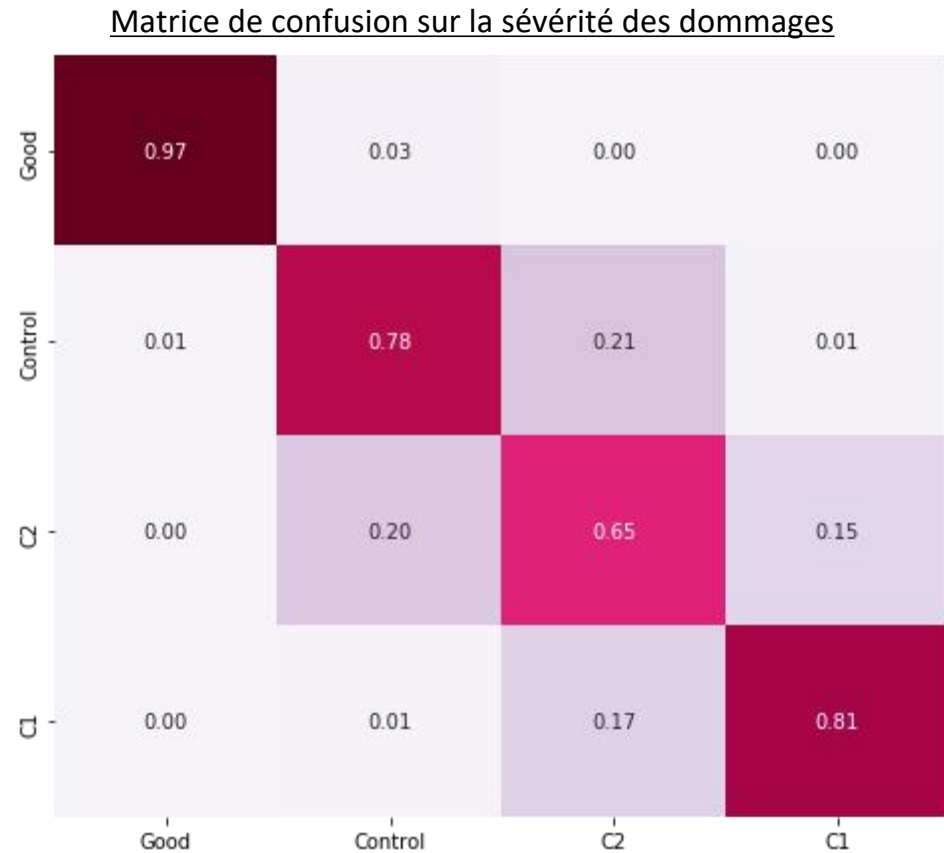
Quand on passe les données au modèle, nous utilisons des **Transformations géométriques** pour faire de la **Data Augmentation**.

- Retournement horizontal
- Retournement vertical
- Rotation
- Rognage aléatoire
- Transposition



Approche 1 : Résultats

Pour un jeu de test contenant 2400 images, nous obtenons les matrices de confusion suivantes :



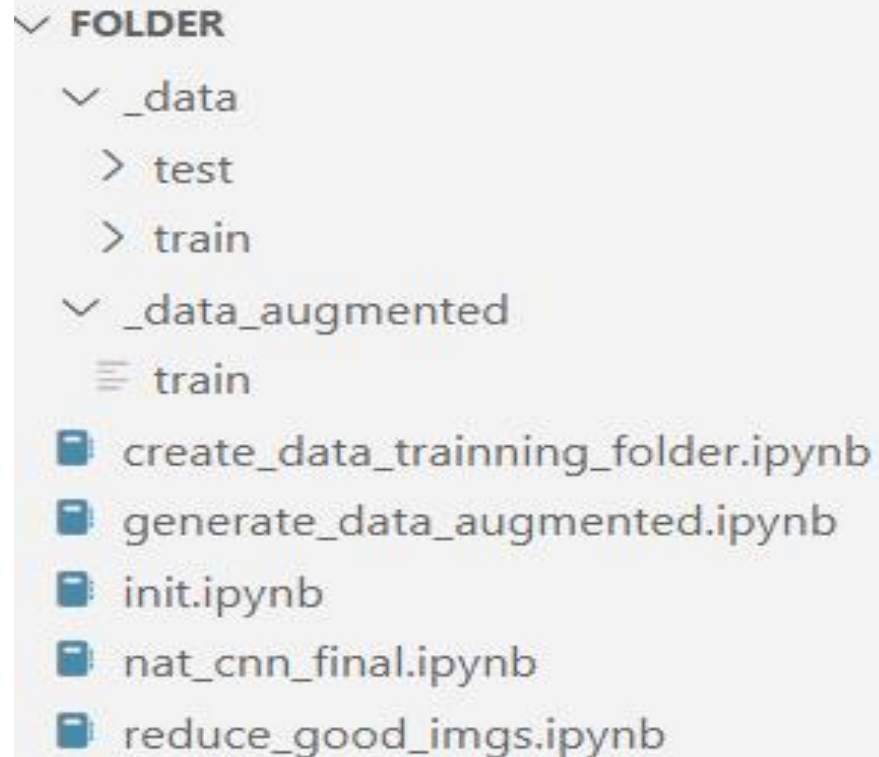
Approche 2

Notre approche à consister effectuer les étapes suivantes :

1. Réorganisation des données de sorte à obtenir 16 classes
2. Pré-augmentation des données
3. Nettoyage des données pré-augmentées
4. Entraînement et évaluation

Approche 2 : Usage de 16 classes

Pour réaliser cette approche, il a fallu passer par une série d'opérations pour aboutir à l'entraînement. C'est ainsi que nous avons défini l'architecture suivante :

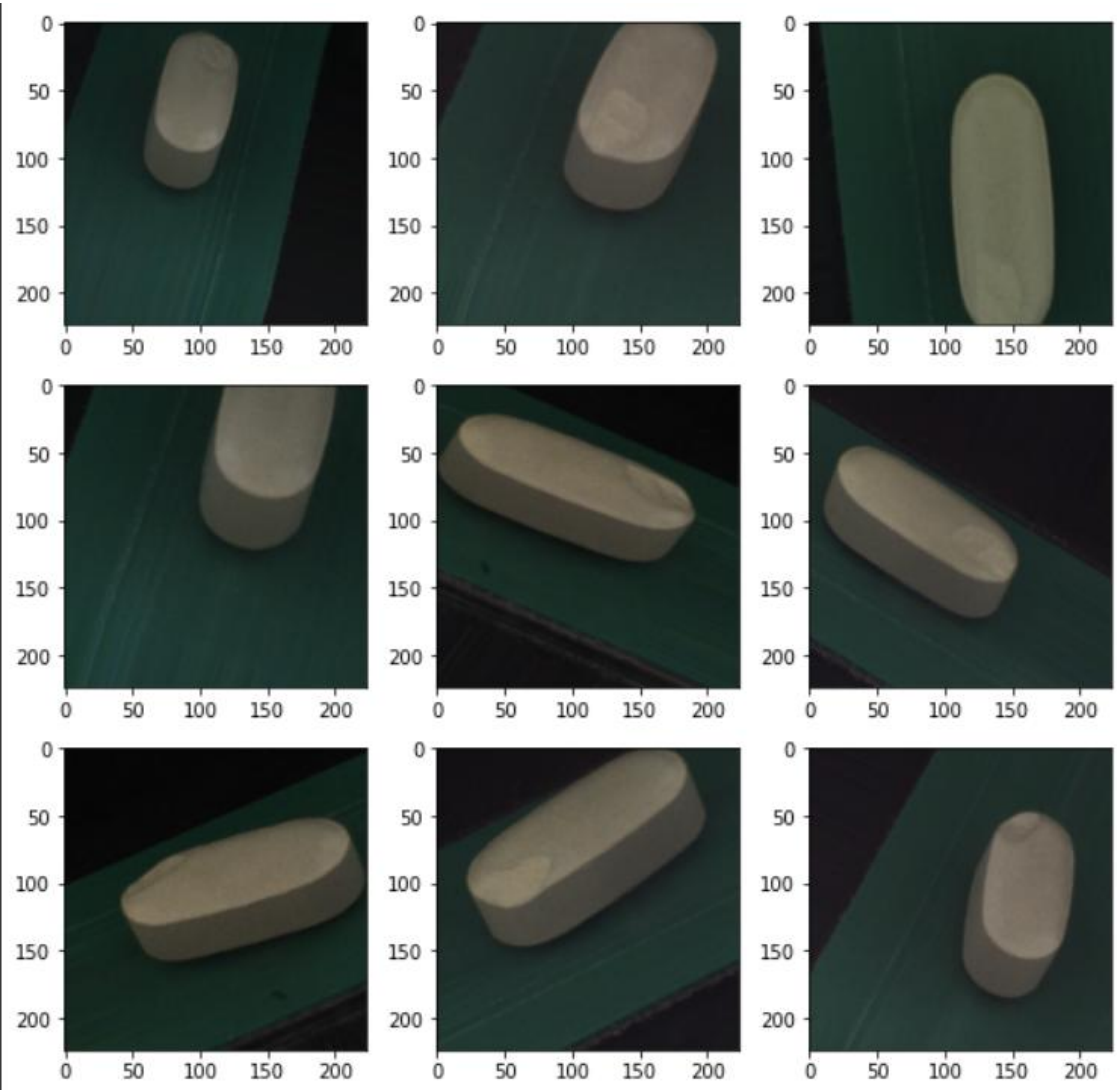


```

✓ FOLDER
  ✓ _data
    > test
    > train
  ✓ _data_augmented
    ≡ train
  create_data_training_folder.ipynb
  generate_data_augmented.ipynb
  init.ipynb
  nat_cnn_final.ipynb
  reduce_good_imgs.ipynb

```

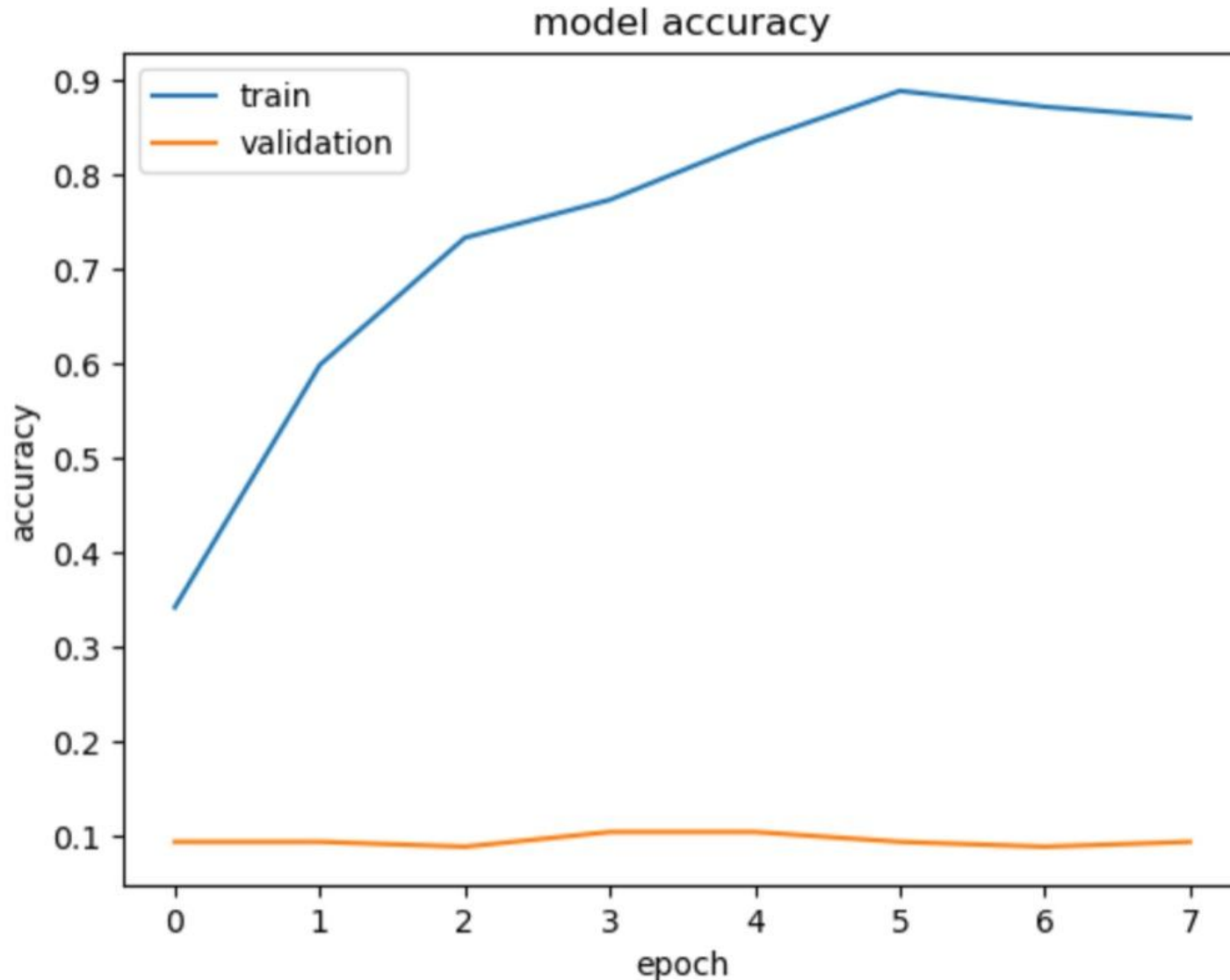

Approche 2 : Pré-augmentation et nettoyage



L'idée générale est d'appliquer les transformations classiques et d'enregistrer l'ensemble des données dans un dossier pour l'entraînement.

L'avantage est de maîtriser et de contrôler processus génération pour avoir l'ensemble de données le plus adapté pour l'entraînement

Approche 2 : Entraînement et résultats



Accuracy: 86%

val accuracy: 9.36%

Conclusions et travaux futurs

Possible improvements:

- Utiliser des **GAN** (Generative Adversarial Network) pour générer de nouvelles données
- Les modèles sont plus ou moins faciles à mettre en place mais la puissance de calcul nécessaire pour les rejeter ou les retenir rapidement faisait défaut.
- En guise d'amélioration, nous pourrions entraîner et tester les modèles avec les 5 ou 10 images de chaque puce mais cela est lié à des contraintes techniques et spécifiques à la machine qui trie et prend les images. Cela pourrait permettre d'obtenir un meilleur réseau neuronal.
- Utilisation d'un modèle siamois pour diminuer les faux négatifs.
- L'utilisation de 16 classes peut réduire la précision et l'overfitting pour ce cas d'utilisation.