Classification d’images

Classification de maladies de peau

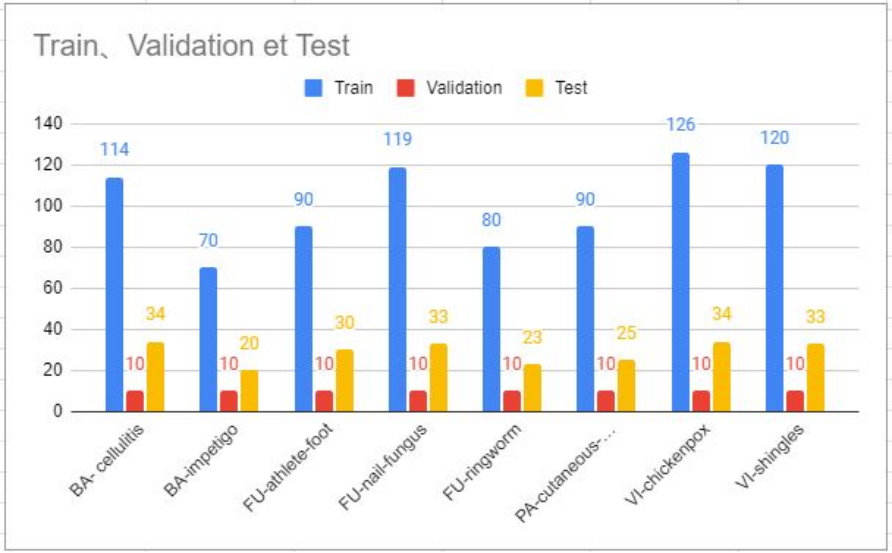
Yingqi LUO

Rétrospective des tâches

- Collecte des données
- Prétraitement des données
- Création de l’ensemble d’entraînement, de validation et de test
- Choix des architectures des modèles
- Entraînement des modèles (Keras et HuggingFace Transformers)
- Évaluation des modèles et compréhension des erreurs
- Optimisation du modèle

Données

- Nature : Photographies médicales
- Source : Kaggle*
- Nombre fichiers totales : 1121
- 8 Classes
  - Infections bactériennes - Cellulite
  - Infections bactériennes - Impétigo
  - Infections fongiques - Pied d'athlète
  - Infections fongiques - Champignon des ongles
  - Infections fongiques - Teigne
  - Infections parasitaires - Larve cutanée migrante
  - Infections virales de la peau - Varicelle
  - Infections virales de la peau - Zona
- 8 dossiers avec des images correspondant aux différents types d'infections.
- Ensembles des données :
  - Entraînement : 809
  - Validation : 80
  - Test : 232
- Format
  - JPG, JPEG, WEBP et PNG
  - Couleurs : RGB et RGBA



Distribution des classes dans les différents sous-ensembles de données du corpus.

*https://www.kaggle.com/datasets/subirbiswas19/skin-disease-dataset/data

Prétraitements sur des données

- Transformation des images au format carré
  - avec une résolution 128 x 128 pour les models MLP et CNN 2D
  - avec une résolution 224 x 224 pour les models ResNet50 (VGG16, ViT)
- Mettre toutes les images au format JPEG avec uniquement RGB.
- Transformer les images en vecteurs de nombre entiers.
- Retrait des couleurs (grayscale) pour les modèles CNN 2D et MLP afin de garder deux dimensions (noir et blanc). Tout le reste des modèles sont en RGB (3 dimensions).

Exemples de classes

Ce projet a pour but de classifier les 8 infections de la peau causées par différents agents pathogènes :

BA-cellulitis

BA-impetigo

FU-athlete-foot

FU-nail-fungus

FU-ringworm

PA-cutaneous-larva-migrans

VI-chickenpox

VI-shingles

Récapitulatif des architectures utilisées

Mes 5 choix d’architectures :

- Multi-Layer Perceptron (MLP)

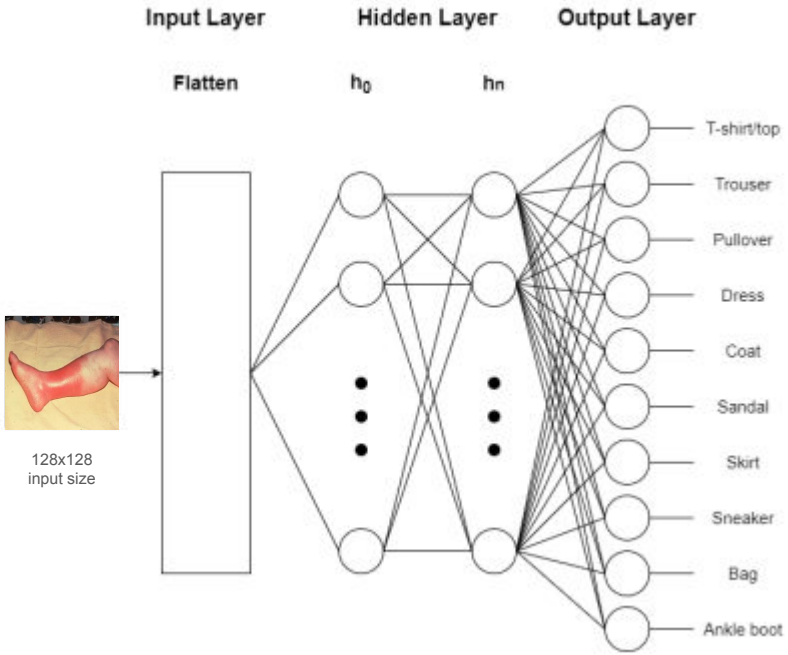
- Convolutional Neural Network (CNN) 2D

- Residual neural network (ResNet 50)

- Visual Geometry Group (VGG 16)

- Vision Transformer (ViT)

MLP - Multi-Layer Perceptron

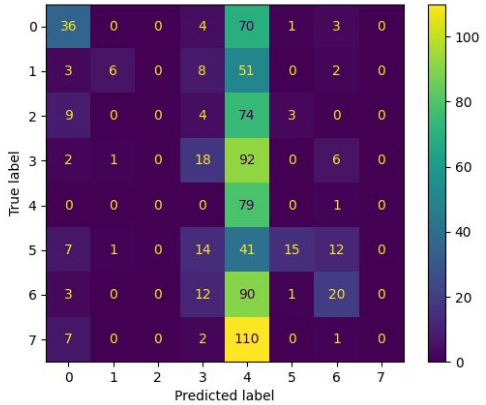



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 4096)              67112960
_________________________________________________________________
dense_1 (Dense)              (None, 256)               1048832
_________________________________________________________________
dense_2 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 520
=================================================================
Total params: 68,178,760
Trainable params: 68,178,760
Non-trainable params: 0
```

MLP à 4 couches - Évaluation et interprétation

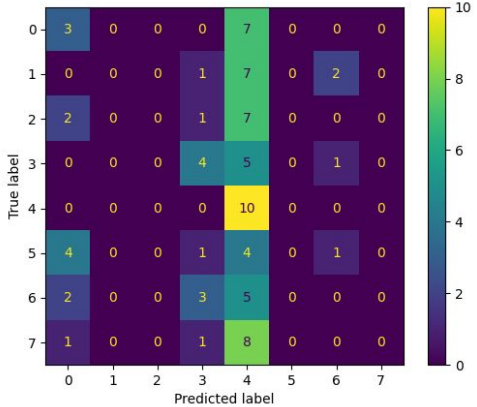
Train

	precision	recall	f1-score	support
0	0.5373	0.3158	0.3978	114
1	0.7500	0.0857	0.1538	70
2	0.0000	0.0000	0.0000	90
3	0.2903	0.1513	0.1989	119
4	0.1301	0.9875	0.2300	80
5	0.7500	0.1667	0.2727	90
6	0.4444	0.1587	0.2339	126
7	0.0000	0.0000	0.0000	120
accuracy			0.2151	809
macro avg	0.3628	0.2332	0.1859	809
weighted avg	0.3488	0.2151	0.1881	809



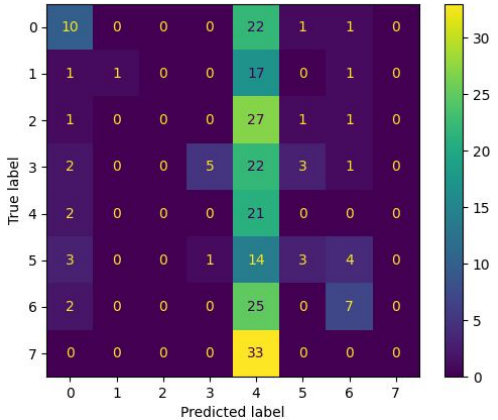
Validation

	precision	recall	f1-score	support
0	0.2500	0.3000	0.2727	10
1	0.0000	0.0000	0.0000	10
2	0.0000	0.0000	0.0000	10
3	0.3636	0.4000	0.3810	10
4	0.1887	1.0000	0.3175	10
5	0.0000	0.0000	0.0000	10
6	0.0000	0.0000	0.0000	10
7	0.0000	0.0000	0.0000	10
accuracy			0.2125	80
macro avg	0.1003	0.2125	0.1214	80
weighted avg	0.1003	0.2125	0.1214	80

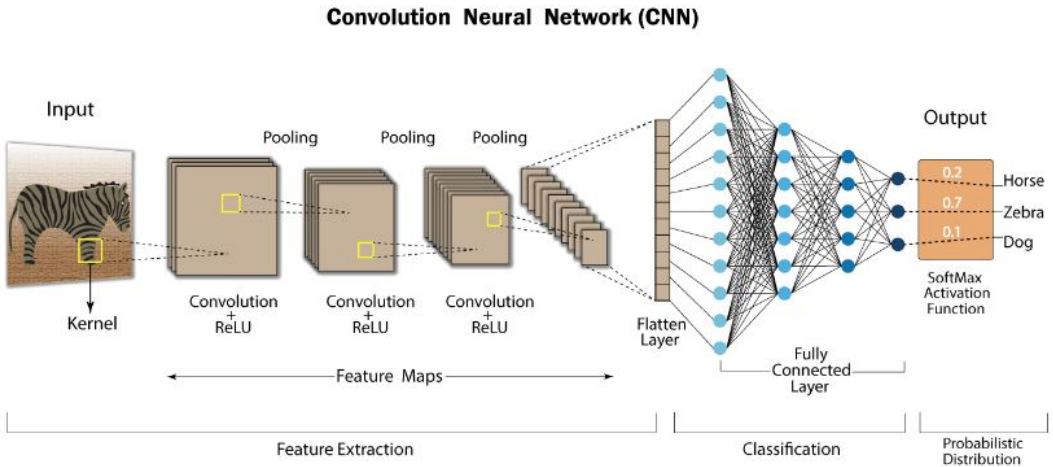


Test

	precision	recall	f1-score	support
0	0.4762	0.2941	0.3636	34
1	1.0000	0.0500	0.0952	20
2	0.0000	0.0000	0.0000	30
3	0.8333	0.1515	0.2564	33
4	0.1160	0.9130	0.2059	23
5	0.3750	0.1200	0.1818	25
6	0.4667	0.2059	0.2857	34
7	0.0000	0.0000	0.0000	33
accuracy			0.2026	232
macro avg	0.4084	0.2168	0.1736	232
weighted avg	0.3948	0.2026	0.1798	232



Convolutional Neural Network (CNN) 2D



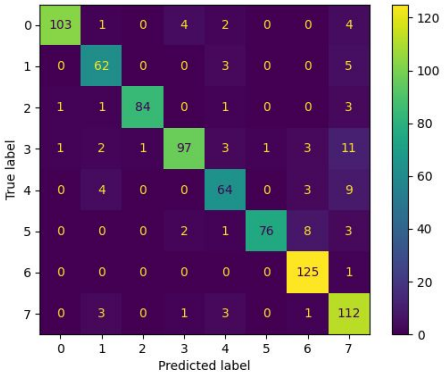

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 128)     3584
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 61, 256)       295168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 256)       590080
_________________________________________________________________
flatten (Flatten)            (None, 200704)            0
_________________________________________________________________
dense (Dense)                (None, 256)               51380480
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 2056
=================================================================
Total params: 52,271,368
Trainable params: 52,271,368
Non-trainable params: 0
```

Kernel : 3x3

CNN 2D - Évaluation et interprétation

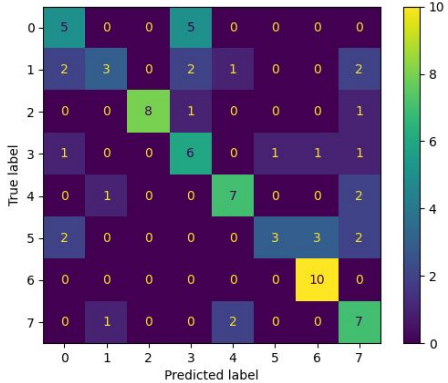
Train

	precision	recall	f1-score	support
0	0.9810	0.9035	0.9406	114
1	0.8493	0.8857	0.8671	70
2	0.9882	0.9333	0.9600	90
3	0.9327	0.8151	0.8700	119
4	0.8312	0.8000	0.8153	80
5	0.9870	0.8444	0.9102	90
6	0.8929	0.9921	0.9398	126
7	0.7568	0.9333	0.8358	120
accuracy			0.8937	809
macro avg	0.9024	0.8884	0.8924	809
weighted avg	0.9022	0.8937	0.8946	809



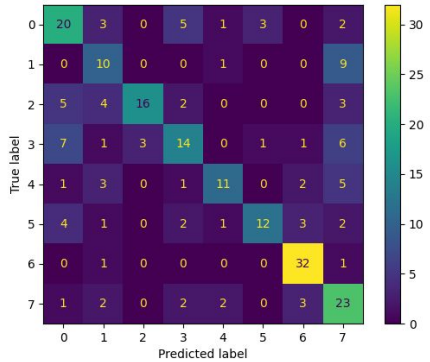
Validation

	precision	recall	f1-score	support
0	0.5000	0.5000	0.5000	10
1	0.6000	0.3000	0.4000	10
2	1.0000	0.8000	0.8889	10
3	0.4286	0.6000	0.5000	10
4	0.7000	0.7000	0.7000	10
5	0.7500	0.3000	0.4286	10
6	0.7143	1.0000	0.8333	10
7	0.4667	0.7000	0.5600	10
accuracy			0.6125	80
macro avg	0.6449	0.6125	0.6013	80
weighted avg	0.6449	0.6125	0.6013	80



Test

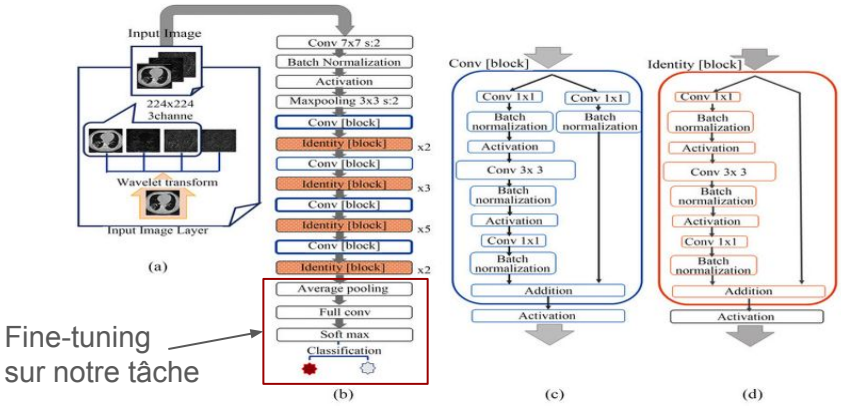
	precision	recall	f1-score	support
0	0.5263	0.5882	0.5556	34
1	0.4000	0.5000	0.4444	20
2	0.8421	0.5333	0.6531	30
3	0.5385	0.4242	0.4746	33
4	0.6875	0.4783	0.5641	23
5	0.7500	0.4800	0.5854	25
6	0.7805	0.9412	0.8533	34
7	0.4510	0.6970	0.5476	33
accuracy			0.5948	232
macro avg	0.6220	0.5803	0.5848	232
weighted avg	0.6246	0.5948	0.5936	232



Residual neural network (ResNet50)

50 couches :

- Convolutional layers
- Pooling layers
- Fully Connected layers



Fine-tuning
sur notre tâche

Nombre de paramètres entraînable:

- Sans pré-apprentissage: trop de paramètre par rapport au nombre de données

```
Total params: 23,604,104
Trainable params: 23,550,984
Non-trainable params: 53,120
```

- Pré-apprentissage sur ImageNet et Fine-tuning sur nos données médicale →

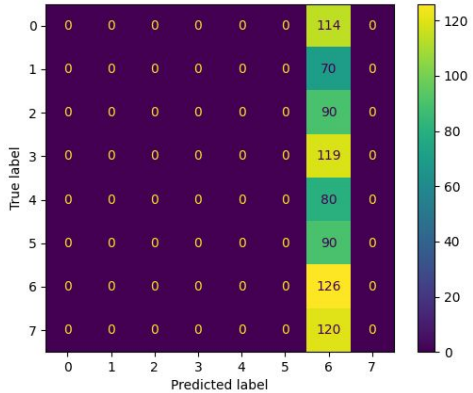
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 batch_normalization (BatchN  (None, 2048)             8192
 ormalization)

 dense (Dense)               (None, 2048)              4196352

 batch_normalization_1 (Batc  (None, 2048)             8192
 hNormalization)

 dense_1 (Dense)             (None, 1024)              2098176

 batch_normalization_2 (Batc  (None, 1024)             4096
 hNormalization)

 dense_2 (Dense)             (None, 8)                 8200

=================================================================
Total params: 29,910,920
Trainable params: 6,312,968
Non-trainable params: 23,597,952
```

ResNet50 (sans préapprentissage) - Évaluation et interprétation

```
Total params: 23,604,104
Trainable params: 23,550,984
Non-trainable params: 53,120
```

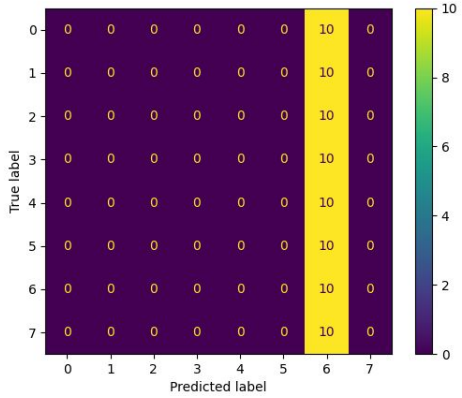
Train

	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	114
1	0.0000	0.0000	0.0000	70
2	0.0000	0.0000	0.0000	90
3	0.0000	0.0000	0.0000	119
4	0.0000	0.0000	0.0000	80
5	0.0000	0.0000	0.0000	90
6	0.1557	1.0000	0.2695	126
7	0.0000	0.0000	0.0000	120
accuracy			0.1557	809
macro avg	0.0195	0.1250	0.0337	809
weighted avg	0.0243	0.1557	0.0420	809



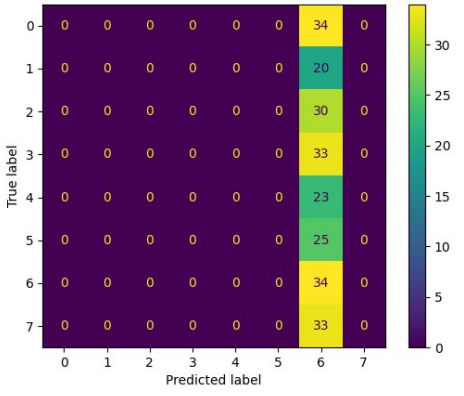
Validation

	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	10
1	0.0000	0.0000	0.0000	10
2	0.0000	0.0000	0.0000	10
3	0.0000	0.0000	0.0000	10
4	0.0000	0.0000	0.0000	10
5	0.0000	0.0000	0.0000	10
6	0.1250	1.0000	0.2222	10
7	0.0000	0.0000	0.0000	10
accuracy			0.1250	80
macro avg	0.0156	0.1250	0.0278	80
weighted avg	0.0156	0.1250	0.0278	80



Test

	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	34
1	0.0000	0.0000	0.0000	20
2	0.0000	0.0000	0.0000	30
3	0.0000	0.0000	0.0000	33
4	0.0000	0.0000	0.0000	23
5	0.0000	0.0000	0.0000	25
6	0.1466	1.0000	0.2556	34
7	0.0000	0.0000	0.0000	33
accuracy			0.1466	232
macro avg	0.0183	0.1250	0.0320	232
weighted avg	0.0215	0.1466	0.0375	232

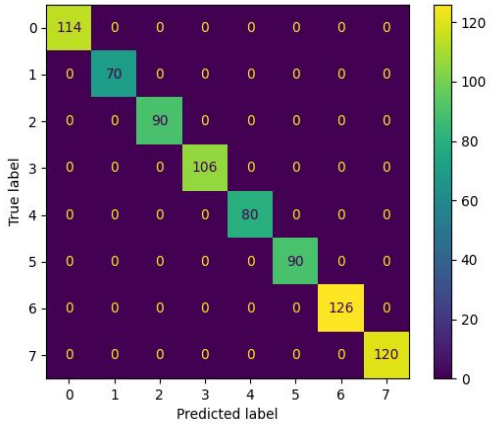


ResNet50 (préapprentissage sur ImageNet et Fine-tuning) - Évaluation et interprétation

```
Total params: 29,910,920
Trainable params: 6,312,968
Non-trainable params: 23,597,952
```

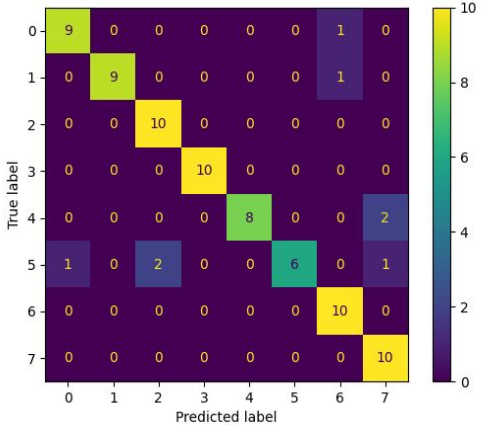
Train

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	114
1	1.0000	1.0000	1.0000	70
2	1.0000	1.0000	1.0000	90
3	1.0000	1.0000	1.0000	106
4	1.0000	1.0000	1.0000	80
5	1.0000	1.0000	1.0000	90
6	1.0000	1.0000	1.0000	126
7	1.0000	1.0000	1.0000	120
accuracy			1.0000	796
macro avg	1.0000	1.0000	1.0000	796
weighted avg	1.0000	1.0000	1.0000	796



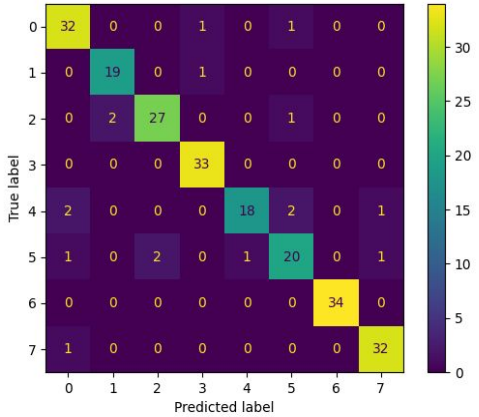
Validation

	precision	recall	f1-score	support
0	0.9000	0.9000	0.9000	10
1	1.0000	0.9000	0.9474	10
2	0.8333	1.0000	0.9091	10
3	1.0000	1.0000	1.0000	10
4	1.0000	0.8000	0.8889	10
5	1.0000	0.6000	0.7500	10
6	0.8333	1.0000	0.9091	10
7	0.7692	1.0000	0.8696	10
accuracy			0.9000	80
macro avg	0.9170	0.9000	0.8968	80
weighted avg	0.9170	0.9000	0.8968	80

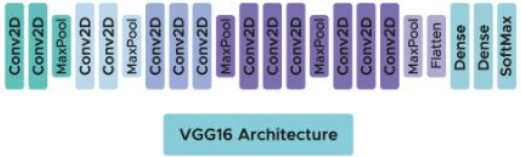


Test

	precision	recall	f1-score	support
0	0.8889	0.9412	0.9143	34
1	0.9048	0.9500	0.9268	20
2	0.9310	0.9000	0.9153	30
3	0.9429	1.0000	0.9706	33
4	0.9474	0.7826	0.8571	23
5	0.8333	0.8000	0.8163	25
6	1.0000	1.0000	1.0000	34
7	0.9412	0.9697	0.9552	33
accuracy			0.9267	232
macro avg	0.9237	0.9179	0.9195	232
weighted avg	0.9269	0.9267	0.9257	232

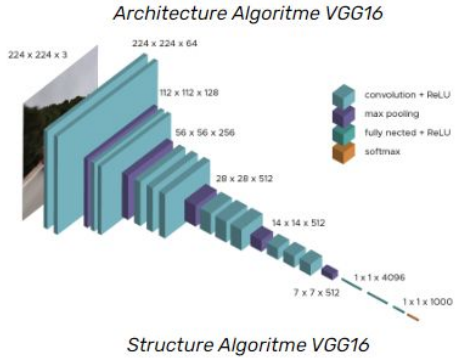


Visual Geometry Group (VGG16)



16 couches :

- Convolutional layers
- Pooling layers
- Fully Connected layers



- Sans pré-apprentissage : trop de paramètre par rapport au nombre de données

```
Total params: 134,293,320
Trainable params: 134,293,320
Non-trainable params: 0
```

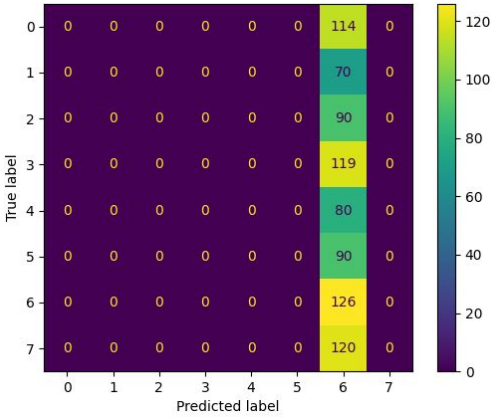
- Pré-apprentissage sur ImageNet et Fine-tuning sur nos données médicale →

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 512)               14714688

 flatten (Flatten)           (None, 512)               0

 batch_normalization (BatchN  (None, 512)              2048
 ormalization)

 dense (Dense)               (None, 2048)              1050624

 batch_normalization_1 (Batc  (None, 2048)             8192
 hNormalization)

 dense_1 (Dense)             (None, 1024)              2098176

 batch_normalization_2 (Batc  (None, 1024)             4096
 hNormalization)

 dense_2 (Dense)             (None, 8)                 8200

=================================================================
Total params: 17,886,024
Trainable params: 3,164,168
Non-trainable params: 14,721,856
```

VGG16 (sans préapprentissage) - Résultats

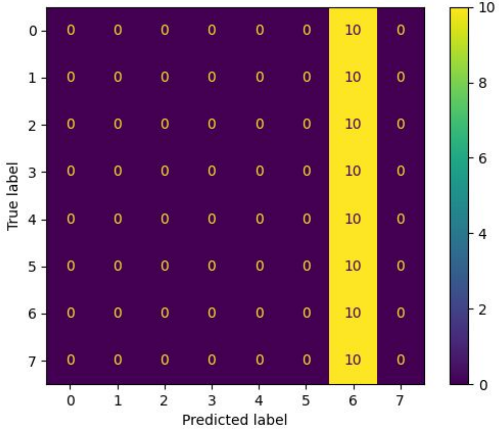
Train

	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	114
1	0.0000	0.0000	0.0000	70
2	0.0000	0.0000	0.0000	90
3	0.0000	0.0000	0.0000	119
4	0.0000	0.0000	0.0000	80
5	0.0000	0.0000	0.0000	90
6	0.1557	1.0000	0.2695	126
7	0.0000	0.0000	0.0000	120
accuracy			0.1557	809
macro avg	0.0195	0.1250	0.0337	809
weighted avg	0.0243	0.1557	0.0420	809



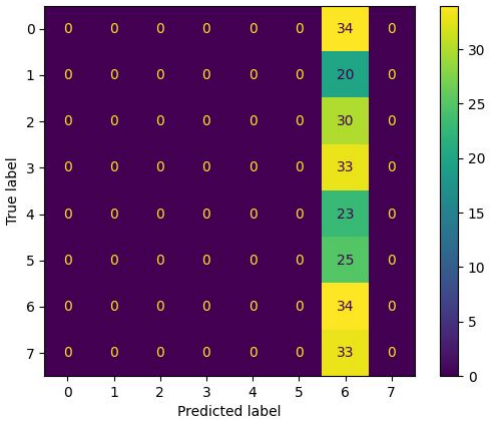
Validation

	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	10
1	0.0000	0.0000	0.0000	10
2	0.0000	0.0000	0.0000	10
3	0.0000	0.0000	0.0000	10
4	0.0000	0.0000	0.0000	10
5	0.0000	0.0000	0.0000	10
6	0.1250	1.0000	0.2222	10
7	0.0000	0.0000	0.0000	10
accuracy			0.1250	80
macro avg	0.0156	0.1250	0.0278	80
weighted avg	0.0156	0.1250	0.0278	80



Test

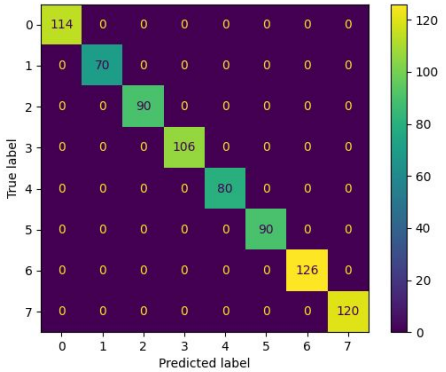
	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	34
1	0.0000	0.0000	0.0000	20
2	0.0000	0.0000	0.0000	30
3	0.0000	0.0000	0.0000	33
4	0.0000	0.0000	0.0000	23
5	0.0000	0.0000	0.0000	25
6	0.1466	1.0000	0.2556	34
7	0.0000	0.0000	0.0000	33
accuracy			0.1466	232
macro avg	0.0183	0.1250	0.0320	232
weighted avg	0.0215	0.1466	0.0375	232



VGG16 (préapprentissage sur ImageNet et Fine-tuning) - Résultats

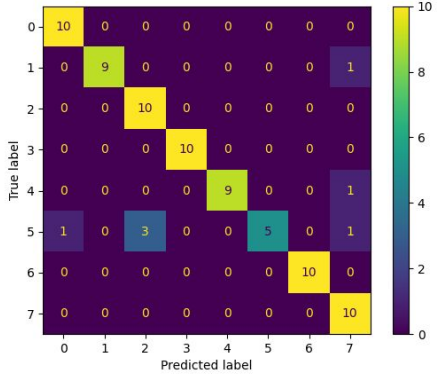
Train

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	114
1	1.0000	1.0000	1.0000	70
2	1.0000	1.0000	1.0000	90
3	1.0000	1.0000	1.0000	106
4	1.0000	1.0000	1.0000	80
5	1.0000	1.0000	1.0000	90
6	1.0000	1.0000	1.0000	126
7	1.0000	1.0000	1.0000	120
accuracy			1.0000	796
macro avg	1.0000	1.0000	1.0000	796
weighted avg	1.0000	1.0000	1.0000	796



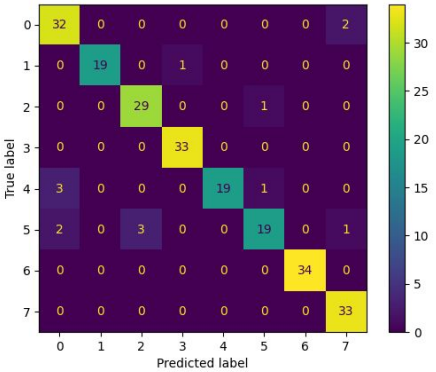
Validation

	precision	recall	f1-score	support
0	0.9091	1.0000	0.9524	10
1	1.0000	0.9000	0.9474	10
2	0.7692	1.0000	0.8696	10
3	1.0000	1.0000	1.0000	10
4	1.0000	0.9000	0.9474	10
5	1.0000	0.5000	0.6667	10
6	1.0000	1.0000	1.0000	10
7	0.7692	1.0000	0.8696	10
accuracy			0.9125	80
macro avg	0.9309	0.9125	0.9066	80
weighted avg	0.9309	0.9125	0.9066	80



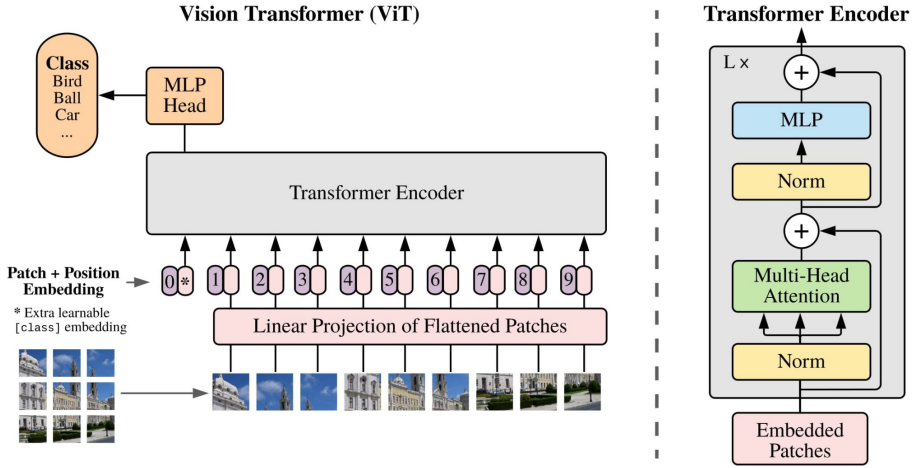
Test

	precision	recall	f1-score	support
0	0.8649	0.9412	0.9014	34
1	1.0000	0.9500	0.9744	20
2	0.9062	0.9667	0.9355	30
3	0.9706	1.0000	0.9851	33
4	1.0000	0.8261	0.9048	23
5	0.9048	0.7600	0.8261	25
6	1.0000	1.0000	1.0000	34
7	0.9167	1.0000	0.9565	33
accuracy			0.9397	232
macro avg	0.9454	0.9305	0.9355	232
weighted avg	0.9418	0.9397	0.9385	232



Vision Transformer (ViT)

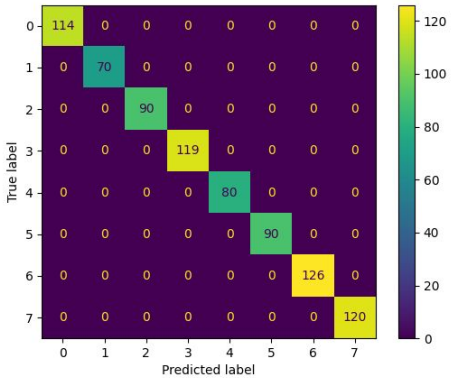
- ViT est un modèle basé sur l'architecture Transformer et plus particulièrement BERT (modèle de traitement du langage naturelle) où seul l'entrée change.
- Il commence part découper les images RGB de dimension 224x224 en morceaux de dimension 16x16.
- Ses morceaux sont ensuite mis à plat (flatten), passés à une couche linéaire puis à la couche embedding du modèle. Afin de donner une représentation vectorielle (embedding) de ces morceaux tout en prenant en compte la position de celui-ci dans l'image.
- Les embeddings sont ensuite utilisés par l'architecture BERT.



ViT (réapprentissage sur ImageNet et Fine-tuning) - Résultats

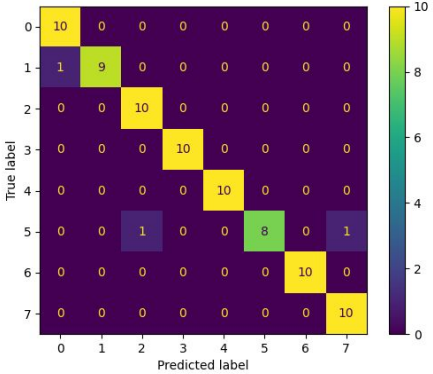
Train

	precision	recall	f1-score	support
BA-cellulitis	1.0000	1.0000	1.0000	114
BA-impetigo	1.0000	1.0000	1.0000	70
FU-athlete-foot	1.0000	1.0000	1.0000	90
FU-nail-fungus	1.0000	1.0000	1.0000	119
FU-ringworm	1.0000	1.0000	1.0000	80
PA-cutaneous-larva-migrans	1.0000	1.0000	1.0000	90
VI-chickenpox	1.0000	1.0000	1.0000	126
VI-shingles	1.0000	1.0000	1.0000	120
accuracy			1.0000	809
macro avg	1.0000	1.0000	1.0000	809
weighted avg	1.0000	1.0000	1.0000	809



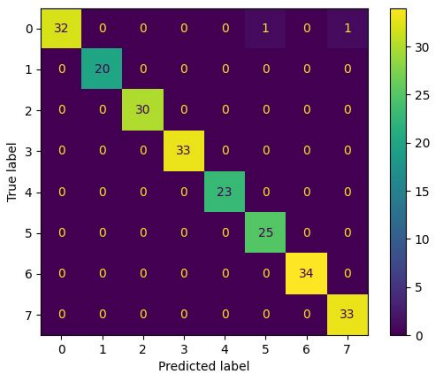
Validation

	precision	recall	f1-score	support
BA-cellulitis	0.9091	1.0000	0.9524	10
BA-impetigo	1.0000	0.9000	0.9474	10
FU-athlete-foot	0.9091	1.0000	0.9524	10
FU-nail-fungus	1.0000	1.0000	1.0000	10
FU-ringworm	1.0000	1.0000	1.0000	10
PA-cutaneous-larva-migrans	1.0000	0.8000	0.8889	10
VI-chickenpox	1.0000	1.0000	1.0000	10
VI-shingles	0.9091	1.0000	0.9524	10
accuracy			0.9625	80
macro avg	0.9659	0.9625	0.9617	80
weighted avg	0.9659	0.9625	0.9617	80



Test

	precision	recall	f1-score	support
BA-cellulitis	1.0000	0.9412	0.9697	34
BA-impetigo	1.0000	1.0000	1.0000	20
FU-athlete-foot	1.0000	1.0000	1.0000	30
FU-nail-fungus	1.0000	1.0000	1.0000	33
FU-ringworm	1.0000	1.0000	1.0000	23
PA-cutaneous-larva-migrans	0.9615	1.0000	0.9804	25
VI-chickenpox	1.0000	1.0000	1.0000	34
VI-shingles	0.9706	1.0000	0.9851	33
accuracy			0.9914	232
macro avg	0.9915	0.9926	0.9919	232
weighted avg	0.9917	0.9914	0.9913	232



Ouverture

- Utiliser un modèle pré-entraîner sur des données **spécialisées** au domaine médical.
- Faire de **l'augmentation** de données (rotation, agrandissement…)
- Au vu de la simplicité des annotations, cela serait intéressant d'**ajouter des classes** supplémentaires.
- Ajouter des images dans le train et test de sorte à **couvrir plus de cas d'usage** (angles, qualités différentes, bruit, luminosité…)

Conclusion

- Le mécanisme d’attention semble fournir des résultats plus intéressants grâce à son architecture plus complexe permettant de se concentrer sur les portions de l’image importantes.
- Les architectures ResNet50 et VGG16 ne parviennent pas à prédire de classes pertinentes quand ils sont entraînés de zéro, ils nécessitent un pré-apprentissage dans le but de mieux reconnaitre les formes.
- Ce comportement est dû à leurs grandes tailles et au fait qu’ils souffrent du manque de données.

Citations

- **ViT**: Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.*" (2021).
- **ResNet50**: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "*Deep Residual Learning for Image Recognition*" (2015).
- **VGG16**: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "*Deep Residual Learning for Image Recognition.*" (2015).