Utilisation du Laplacien pour le filtrage du dataset.

1. Contexte

Dans notre situation, nous avons un jeu de données de radios pulmonaires, et nous souhaiterions filtrer ledit jeu de données afin de se défausser des image floues ou de mauvaises qualités.

Nous utiliserons pour cela la variance de l'opéateur Laplacien, dont le fonctionnement est le suivant :

- **Fondements Mathématiques :**

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 2: The Laplacian kernel (LK)

L'idée est de faire une convolution matricielle entre LK et la matrice X de l'image (X doit être au format grayscale, ce qui est notre cas), ce qui approxime le gradient (soit une matrice combinant l'ensemble des dérivées partielles de chaque point) et de prendre la variance du résultat. Si cette variance tombe en-dessous d'un certain seuil « *threshold* », l'image est considérée comme floutée.

En effet, à travers LK on calcule la dérivée 2nde de l'image

(dérivée discrète, de par la nature des pixels, donc on applique deux fois $\lim_{k\to 0}(f(n+k)-f(n))/k$),

ce qui permet de mettre en exergue les variations d'intensité lumineuses au sein de l'image, et donc, les images d'avantage porteuses d'information. Donc à partir de là, deux cas de figure :

_ la variance est relativement élevée, donc la dérivée 1ère varie beaucoup, donc l'image possède beaucoup de parties informationnelles ;

_ la variance est relativement faible, donc la dérivée 1ère varie peu, donc l'image possède peu de parties informationnelles.

De par la nature de cette méthode, le seuil *threshold* choisi pour la technique du Laplacien dépendra de l'environnement choisi.

N.B : le filtre de Shobël peut également être utilisé pour détecter la significativité d'une matrice. On utilisera dans ce cas non pas une mais deux matrices, dont on prend la racine carré de la somme élevée au carré pour approximer le gradient (et le reste fonctionne comme l'opérateur Laplacien).

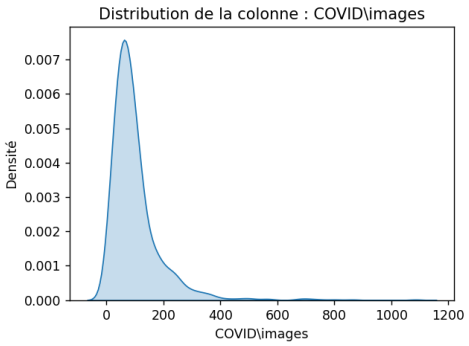
- **Code Python**

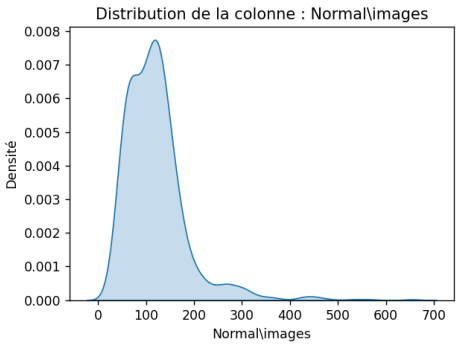
En pratique, plutôt que de recréer l'opérateur Laplacien «à la main», on utilisera la librairie *opencv* via `cv2.Laplacian(gray, cv2.CV_64F)`, où *gray* repésente notre image, et *cv2.CV_64F* représente la profondeur *ddepth* (on demande d'enregistrer les pixels comme nombre flottants sur 64 bits, étant donné que le Lapla va effectuer l'équivalent matriciel des dérivées secondes).

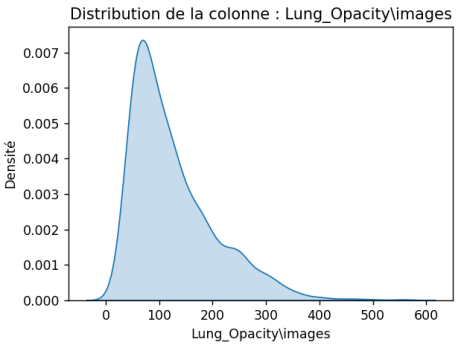
2. Statistiques du Laplacien

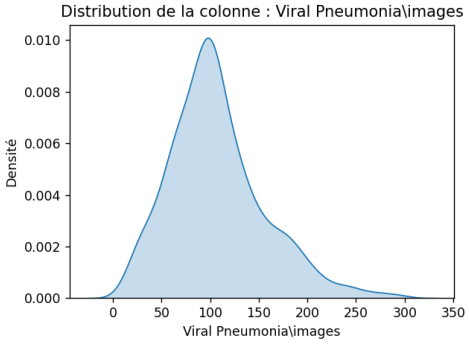
On étudiera tout d'abord les statistiques de la variance du Laplacien sans poser de seuil de présélection :

	index	COVID	Normal	Lung_Opacity	Viral Pneumonia
0		101,038			
		8266400	119,6851	125,67878173722	106,70079208378
	mean	57	1830232	8	7
1		76,1288	111,7625		
		3062411	1872324	103,68651129642	100,75754893979
	median	55	2	5	9
2		96,8558	66,7680		
		9723388	0750569	78,188438345471	48,930626644141
	std	7	6	7	9
3		5,56371	27,7968		
		7748662	6788811	22,144093081394	10,570326991897
	min	83	12	4	7
4		1085,68	654,698		
		6870221	7924434	557,95466581132	295,87550223217
	max	8	35	5	2







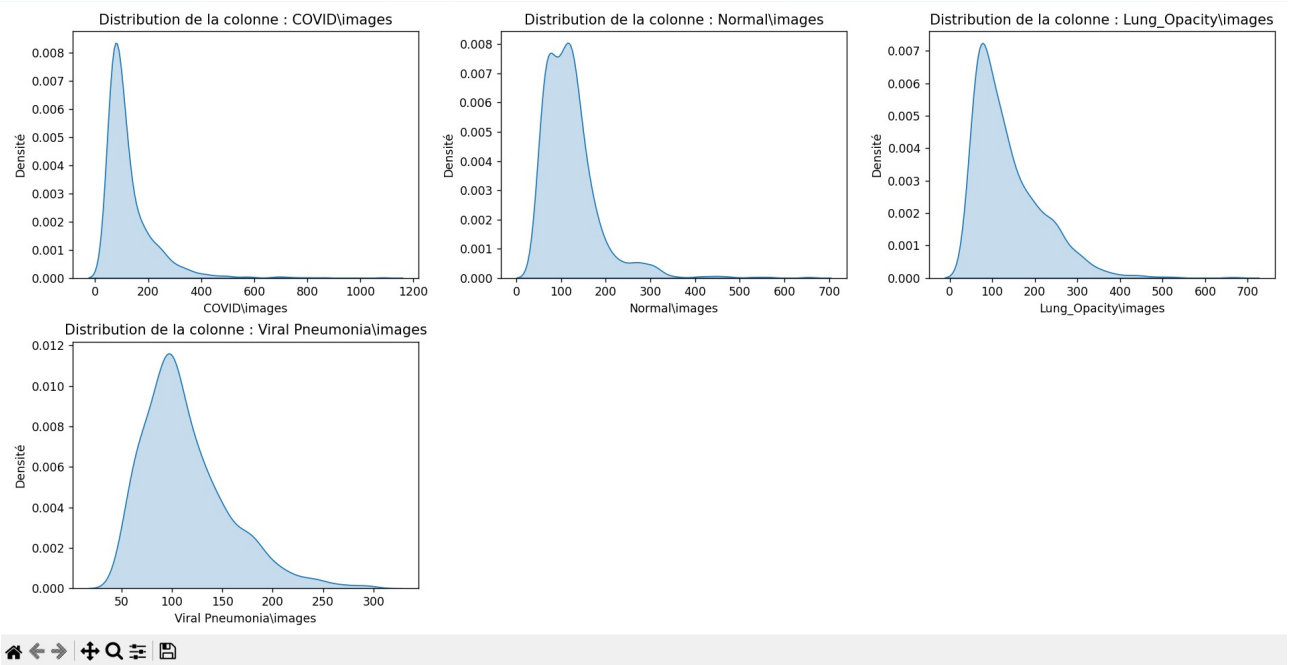


On se rend compte d'au moins deux choses ici :

- La moyenne de la variance pour chaque catégorie est de l'ordre de 100 et quelques, et le plus petit écart-type est d'un peu moins de 50 pour **Vir_Pneum.** On choisira donc un seuil de : **threshold = 100 – 50 = 50.**
- La répartition de la variance est assez dispersée, mais toutes les catégories mise à part **Vir_Pneum** souffrent d'un déséquilibre de la moyenne de leur variance laplacienne de 20 au-dessus de leur médiane, ce qui indique la présence d'outliers.

Après application du Laplacien, on obtient les statistiques suivantes :

	index	COVID	Normal	Lung_Opacity	Viral Pneumonia
0		128,201	121,506		
		8608812	2932912	137,384556373	
	mean	55	24	187	113,964496101606
1		95,2103	111,7682		
		2247060	3569905	113,0305963720	103,78696549264
	median	09	4	78	3
2		99,1488	64,5395		
		8364674	2456191	79,9229636401	
	std	74	77	64	42,892246781204
3		50,0835	50,0292		
		8073631	0214212	50,3167950931	50,601843708209
	min	33	42	54	2
4		1085,68	654,698		
		6870221	7924434	665,142920877	295,87550223217
	max	8	35	263	2
5	Images_ava				
	nt	3616	10192	6012	1345
6	Images_aprè				
	s	2758	9046	5445	1209
7		76,2721	88,7558		
		2389380	8697017	90,5688622754	89,888475836431
	% gardé	53	27	491	2



On se rend compte que :

- Les statistiques des différents échantillons se sont quasiment toutes rapprochées les unes des autres, uniformisant statistiquement un peu plus les différentes classes, mis à part pour l'écart-type de **Vir_Pneum.** dû à une moins grande amplitude de ses outliers. Néanmoins, en lui supprimant ses images de mauvaise qualité, nous avons rapprocher sa distribution de celle des autres classes.
- Nous avons conservé entre 76% et 90% du jeu d'origine, ce qui est bien pour ne pas accroître le déséquilibre de classes.

Nous resterons donc sur notre seuil **threshold = 50.**

3. Comparaison des métriques

Pour évaluer l'impact de l'utilisation du Laplacien sur notre jeu de données, nous utiliserons un CNN convolutif simple (d'abord sur des **échantillons de 100 images/catégorie**) : MobileNet. Nous regarderons le rapport de classification dudit modèle sur le dataset de base et sur le dataset reformaté, en faisant attention spécifiquement à l'*accuracy* (exactitude de notre modèle) et au *recall* (rappel de notre modèle), que nous privilégierons au score de précision.

En effet, puisque l'objectif est la détection de maladies pulmonaires dans un but de prévention, nous préfererons avoir des erreurs de type I (faux positifs) que des erreurs de type II (faux négatifs). En d'autres termes, il vaut mieux avoir des personnes saines diagnostiquées malades que des personnes malades diagnostiquées saines (quitte à produire des examens médicaux pour confirmer la pathologie).

Attention : l'utilisation de MobileNet ne présume rien de définitif dans le projet, et n'est utile que pour anticiper l'impact de l'utilisation du Laplacien sur notre modèle final. Une comparaison des modèles et une sélection des hyperparamètres optimaux seront nécessaires a posteriori.

- **Métriques sur le dataset de base (n=100) :**

```
Accuracy: 0.6625
Recall: 0.6625
Classification Report:
                precision    recall  f1-score   support

         COVID       0.90      0.45      0.60        20
   Lung_Opacity       0.58      0.90      0.71        20
        Normal       0.64      0.35      0.45        20
Viral_Pneumonia       0.68      0.95      0.79        20

      accuracy                           0.66        80
     macro avg       0.70      0.66      0.64        80
  weighted avg       0.70      0.66      0.64        80

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

- **Métriques sur le dataset reformaté (n=100;t=50) :**

```
Accuracy: 0.6750
Recall: 0.6750
Classification Report:
                precision    recall  f1-score   support

         COVID       0.58      0.75      0.65        20
   Lung_Opacity       0.65      0.75      0.70        20
        Normal       0.80      0.20      0.32        20
Viral_Pneumonia       0.77      1.00      0.87        20

      accuracy                           0.68        80
     macro avg       0.70      0.68      0.63        80
  weighted avg       0.70      0.68      0.63        80

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

Nous observons une *accuracy* et un *recall légèrement* plus élevés sur le dataset filtré que sur le dataset original, mais la taille de l'échantillon (400 images = 2% du dataset) étant relativement faible et les germes (*seeds*) aléatoires non-fixés, nous décidons de fixer ces derniers, et de passer à des échantillons de **1000 images/catégorie** (= 20% du dataset) afin d'obtenir des résultats plus significatifs.

- **Métriques sur le dataset de base (n=1000;random.seed=1234) :**

```
Accuracy: 0.8688
Recall: 0.8687
Classification Report:
                precision    recall  f1-score   support

         COVID       0.84      0.90      0.87       200
   Lung_Opacity       0.90      0.83      0.87       200
        Normal       0.84      0.76      0.80       200
Viral_Pneumonia       0.89      0.98      0.94       200

      accuracy                           0.87       800
     macro avg       0.87      0.87      0.87       800
  weighted avg       0.87      0.87      0.87       800

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

- **Métriques sur le dataset filtré (n=1000;t=50;random.seed=1234) :**

```
Accuracy: 0.8562
Recall: 0.8562
Classification Report:
                precision    recall  f1-score   support

         COVID       0.80      0.89      0.84       200
   Lung_Opacity       0.88      0.79      0.83       200
        Normal       0.83      0.77      0.80       200
Viral_Pneumonia       0.92      0.98      0.95       200

      accuracy                           0.86       800
     macro avg       0.86      0.86      0.85       800
  weighted avg       0.86      0.86      0.85       800

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

On se rend compte que le seuil de 50 pour la variance du Laplacien a fait décroître les performances du modèle. On essayera donc des valeurs inférieures.

- **Métriques sur le dataset filtré (n=1000;t=45;random.seed=1234) :**

```
Accuracy: 0.8350
Recall: 0.8350
Classification Report:
                precision    recall  f1-score   support

         COVID       0.70      0.93      0.80       200
   Lung_Opacity       0.93      0.69      0.80       200
        Normal       0.90      0.72      0.80       200
Viral_Pneumonia       0.89      0.99      0.94       200

      accuracy                           0.83       800
     macro avg       0.85      0.83      0.83       800
  weighted avg       0.85      0.83      0.83       800

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

- **Métriques sur le dataset filtré (n=1000;t=40;random.seed=1234) :**

```
Accuracy: 0.8773
Recall: 0.8775
Classification Report:
                precision    recall  f1-score   support

         COVID       0.95      0.78      0.85       200
   Lung_Opacity       0.75      0.96      0.84       200
        Normal       0.89      0.78      0.83       200
Viral_Pneumonia       0.97      0.99      0.98       199

      accuracy                           0.88       799
     macro avg       0.89      0.88      0.88       799
  weighted avg       0.89      0.88      0.88       799

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

On constate une amélioration par rapport au dataset non-filtré. Essayons d'abaisser encore le seuil de présélection afin de voir si nous pouvons encore améliorer les métriques.

- **Métriques sur le dataset filtré (n=1000;t=35;random.seed=1234) :**

```
Accuracy: 0.8725
Recall: 0.8725
Classification Report:
                precision    recall  f1-score   support

         COVID       0.90      0.82      0.86       200
   Lung_Opacity       0.85      0.87      0.86       200
        Normal       0.84      0.81      0.82       200
Viral_Pneumonia       0.90      0.99      0.94       200

      accuracy                           0.87       800
     macro avg       0.87      0.87      0.87       800
  weighted avg       0.87      0.87      0.87       800

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

- **Métriques sur le dataset filtré (n=1000;t=30;random.seed=1234) :**

```
Accuracy: 0.8788
Recall: 0.8788
Classification Report:
                precision    recall  f1-score   support

         COVID       0.93      0.79      0.86       200
   Lung_Opacity       0.86      0.86      0.86       200
        Normal       0.81      0.88      0.84       200
Viral_Pneumonia       0.93      0.98      0.95       200

      accuracy                           0.88       800
     macro avg       0.88      0.88      0.88       800
  weighted avg       0.88      0.88      0.88       800

PS C:\Users\ambro\Desktop\Projet Metier\Code_projet>
```

C’est dont avec un seuil t=30 que nous observons la meilleure amélioration des métriques de notre modèle, et c’est donc celui que nous conseillons de conserver pour la suite du projet.

Statistiques du rééchantillonage :

	index	COVID	Normal	Lung_Opacity	Viral Pneumonia
0		111,64798390842	109,2579702925	126,902976680817	108,053464228064
	mean	5	26		
1		88,97018435606	89,21176023803	104,283534698101	101,135915533731
	median	84	45		
2		82,51854583932	69,01794118186	78,5594713994762	44,9709320503383
	std	67	82		
3		30,03104136222	30,16481343848	31,156925956849	30,1600578108973
	min	26	28		
4		1085,686870221	715,0704188692	665,142920877263	295,875502232172
	max	8	75		
5	Images_avant	3616	10192	6012	1345
6	Images_après	3217	10142	5996	1298
7		88,96570796460	99,50941915227	99,7338656021291	96,5055762081784
	% gardé	18	63		

4. <u>Suggestions d’intégration et conclusion</u>

Nous avons constaté, grâce au filtrage du jeu de données via l’opérateur différentiel Laplacien, une amélioration des métriques que nous cherchons à optimiser dans un contexte de détection de pathologies (à savoir l’*accuracy* et le *recall*), et ce pour seulement une faible réduction du dataset original (entre 11% et moins de 1%), et ce, en utilisant un CNN «naïf» sans aucun travail effectué pour l’optimiser à nos données

Cela présume que nous pourrions obtenir un gain de performances bien plus grand en couplant le filtrage par Laplacien à d’autres méthodes dans les phases de preprocessing et de modélisation, notamment :

- **Autoencoder,**
- **PCA avec EigenImages,**
- **Suppression des doublons,**
- **Méthodes d’Oversampling et d’UnderSampling,**
- **Optimisation des hyperparamètres (GridSearchCV, BayesSearchCV),**
- **etc.**