

TP2: Filtrage d'images et apprentissage profond

GBM8770 – Automne 2024

Professeure: Farida Cheriet

Chargés de laboratoire : Zacharie Legault et Emmanuelle Richer

Objectifs : Ce laboratoire se compose de deux parties. La première partie porte sur le filtrage spectral des images et la deuxième partie sur l'utilisation de l'apprentissage profond sur des images médicales.

Remise du travail : La date de remise est le 7 novembre à 23h30. Une pénalité de 3 points par jour sera appliquée lors d'un retard.

Documents à remettre : Le code et les réponses aux questions sont à compléter dans les notebooks IPython fournis avec cet énoncé (`tp2_seance1.ipynb` et `tp2_seance2.ipynb`).

Le dossier contenant le notebook et toutes les ressources nécessaires à son exécution sont à remettre sous la forme d'une archive ZIP nommée

GBM8770_TP2_équipe_Nom_matricule_Nom_matricule

(par exemple : **GBM8770_TP2_0_Richer_1234567_Legault_7654321**).

Commentez votre code ! Lorsque votre code (et donc vos résultats) est incorrect, vos commentaires peuvent nous permettre de valoriser votre approche...

! Attention! Nous avons toléré certaines mauvaises pratiques dans le TP1 qui ne seront pas acceptées à l'avenir (et nous nous permettrons d'enlever des points).

- N'utilisez pas de double boucle **for** pour itérer sur les pixels d'une image : utilisez les fonctions vectorisées de **numpy**.
- Lorsque vous affichez une image en tons de gris, spécifiez une colormap appropriée (typiquement **cmap="gray"**).

Séance I.

Filtrage fréquentiel 2D

i Nous vous conseillons d'utiliser la plateforme de traitement d'image présentée par Clément Playout dans le cours, particulièrement le module sur la transformée de Fourier 2D. Vous pouvez vous-même créer des signaux sinusoïdaux (en cliquant sur le bouton Draw sin(x)), changer l'angle et la fréquence du signal, et visualiser le résultat de la FFT 2D.

Exercice I : FFT de signaux théoriques 2D

Soit le signal $S_1(x, y) = \cos(2\pi(xf_x + yf_y))$ paramétré par les fréquences f_x et f_y , échantillonné à une fréquence de 100 px mm^{-1} . La fonction **S1(fx, fy)** est déjà implémentée.

Q1. Implémentez la fonction **compute_fft2(signal)** qui prend en argument une image et calcule sa transformée de Fourier (normalisée et avec son origine centrée).

En prenant $f_x = 13 \text{ mm}^{-1}$ et $f_y = 0$, affichez le signal $S_1(x, y)$ et son spectre à l'aide de **plot_fft2()**. Vérifiez la position des pics et leur amplitude (donnée par la colorbar). Les valeurs des positions et d'amplitude sont-elles celles attendues?

i Vous pouvez utiliser la fonction **np.fft.fftshift** pour centrer le spectre de votre transformée de Fourier.

i Le concept de normalisation dans les fonctions de la transformée de Fourier rapide (FFT) peut parfois prêter à confusion. Ce que cela signifie, en pratique, c'est que l'on souhaite que l'amplitude du spectre de la FFT d'un signal de magnitude unitaire (par exemple $s(t) = \sin(2\pi * f * t)$) reste elle aussi unitaire. Il est donc essentiel de bien comprendre l'implémentation de la transformée de Fourier utilisée et de s'adapter en conséquence.

La fonction `np.fft.fft2` dans NumPy applique une normalisation particulière (voir la documentation pour le module `np.fft` et la fonction `np.fft.fft2`). Par défaut, la fonction entraînera une multiplication des valeurs du spectre par le nombre d'éléments dans le signal. Vous pouvez soit diviser le spectre par la valeur appropriée, soit ajuster directement la normalisation avec le paramètre `fft2(..., norm=...)`.

Vous pouvez également aller regarder le notebook [Normalisation.ipynb](#) fourni avec les fichiers de ce TP pour mieux comprendre l'utilité de la normalisation et son implémentation.

! Prenez le temps de lire la documentation de la fonction `plot_fft2(signal)` et de bien spécifier les arguments appropriés lorsque vous l'utilisez.

Q2. Affichez S_1 et son spectre pour $f_x = 0$ et $f_y = 13 \text{ mm}^{-1}$ puis pour $f_x = 5 \text{ mm}^{-1}$ et $f_y = 12 \text{ mm}^{-1}$. Quel est l'effet d'une rotation de l'image sur son spectre ?

On étudie maintenant le signal : $S_2(x, y) = \cos(2\pi f r)$ où $r = \sqrt{x^2 + y^2}$.

Q3. Implémentez `S2(f)`, puis affichez le signal et son spectre pour $f = 20 \text{ mm}^{-1}$ et pour $f = 40 \text{ mm}^{-1}$. Quel est l'effet d'un rétrécissement de l'image sur son spectre ? Affichez le signal et le spectre pour $f = 120 \text{ px mm}^{-1}$. Expliquez l'allure du spectre et les aberrations visibles sur le signal.

i Fiez-vous à l'implémentation donnée de la fonction `S1(fx, fy)`. Allez regarder la documentation de `np.meshgrid`.

Exercice II : Filtrage spectral

Cet exercice étudie une image angiographique du réseau coronaire : c'est-à-dire une radiographie des artères qui alimentent le coeur dans lesquelles est injecté un agent de contraste.

Q1. Implémentez la fonction `gaussian(std, size, x0=0, y0=0)` qui renvoie une matrice carrée de taille `size × size` contenant les valeurs d'une gaussienne d'écart-type `std` et centrée sur `(x0, y0)`. Calculez une gaussienne centrée, d'écart-type 7 et de taille 300×300 . Affichez son image et son spectre.

Q2. Chargez l'image `angiographie_bruit.png`. Affichez l'image et son spectre. Identifiez sur le spectre les raies qui sont responsables des rayures diagonales sur l'angiographie (donnez les coordonnées des points concernés).

Q3. À l'aide de la fonction `gaussian` implémentée précédemment, concevez un masque pour filtrer ces raies directement dans le domaine de Fourier (on pourra prendre un écart type de 3 pixels pour les gaussiennes). Appliquez le masque à la transformée de fourier de l'image. Affichez le masque et le spectre filtré.

i Vous pourriez décider de filtrer uniquement les raies dominantes, ou également les harmoniques. À vous de choisir.

Q4. Implémentez la fonction `compute_ifft2(fft_signal)` qui prend en argument la transformée de Fourier d'une image et renvoie l'image reconstituée par transformée de Fourier inverse. Reconstituez et affichez l'angiographie nettoyée de ses rayures diagonales.

i N'oubliez pas la normalisation dans la `fft` et son effet inverse dans la `ifft`. De plus, la fonction `np.fft.ifft2` retourne une matrice complexe, à vous de récupérer uniquement les valeurs qui vous intéressent.

Pour la suite de l'exercice on travaillera sur l'angiographie nettoyée ou sur son spectre. Sa transformée de Fourier sera notée T_0 .

Q5. On souhaite appliquer un filtre passe-bas dont la réponse fréquentielle est une gaussienne centrée sur l'origine d'écart-type 25. Créez le masque gaussien correspondant, appliquez-le à la transformée de Fourier de l'image et reconstituez-la. Affichez l'image reconstituée et son spectre.

On notera T_{LF} la transformée de Fourier après ce filtrage.

Q6. Calculez l'intensité des fréquences qui ont été retirées du spectre à la question précédente, c'est-à-dire la transformée de Fourier T_{HF} telle que $T_0 = T_{LF} + T_{HF}$. Affichez l'image reconstituée et son spectre. Quel est le type de ce filtrage ?

Q7. Créez un masque gaussien centré d'écart-type 1.91 px, de taille 11×11 et normalisé pour que sa somme soit 1. Convoluez ce masque avec l'image nettoyée à la question 4. Affichez l'image filtrée et son spectre.

Q8. Démontrez mathématiquement l'équivalence entre les filtrages réalisés aux questions 5 et 7. Avec quel masque faut-il convoluer l'image nettoyée pour réaliser un filtrage passe-haut équivalent à celui de la question 6 mais dans le domaine spatial ?

i On rappelle que la transformée de Fourier d'une gaussienne d'écart-type σ_0 est une gaussienne d'écart-type $\frac{1}{2\pi\sigma_0}$:

$$h(x) = \exp\left(\frac{-x^2}{2\sigma^2}\right) \xrightarrow{\text{Fourier}} H(f) = \sigma\sqrt{2\pi} \exp\left(-2(\pi\sigma f)^2\right)$$

Notons aussi que l'écart-type du masque de la question 5 en px^{-1} est de $\frac{25}{300}$, c'est-à-dire l'écart-type de la gaussienne divisé par le nombre de pixel de la largeur (ou de la hauteur) du spectre.

Q9. (Bonus) Calculez et affichez la transformée de Fourier de l'image **cellules.png**. Cette image représente une certaine lignée cellulaire imagée au microscope. Est-ce que la méthode implémentée dans cet exercice permettrait d'éliminer les lignes qui perturbent l'image ? En quoi le spectre de la FFT est-il différent de celui de l'image angiographique ?

? Vous pouvez convertir, d'une façon pertinente, l'image **cellules.png** (qui possède 4 canaux) à une image en tons de gris (1 seul canal), puis utiliser les fonctions préalablement réalisées pour afficher l'image et son spectre.

Séance II.

Apprentissage profond appliqué aux images médicales

Les prochains exercices vous guideront au travers des bases de l'entraînement de réseaux de neurones.

! Attention ! Les réseaux de neurones peuvent facilement consommer beaucoup de ressources de votre ordinateur. Si votre ordinateur n'est pas muni d'un GPU dédié (idéalement de marque Nvidia), nous vous déconseillons de faire le TP sur votre propre ordinateur. Les ordinateurs du local L-4818 sont munis de GPU appropriés, il serait mieux de les utiliser dans ce cas.

Sinon, vous pouvez exécuter le notebook sur Google Colab, où vous aurez accès gratuitement à un GPU. Soyez cependant conscients que Colab ne vous permet que quelques heures d'utilisation avant que vous perdiez l'accès à un GPU. Aussi, si vous perdez votre connexion Internet, l'environnement se déconnectera et vous perdrez vos variables.

Si utilisez les ordinateurs du labo, suivez les instructions suivantes. Sinon, le notebook dans Google Colab fera les installations nécessaires.

i Les lignes de code fournies ici assument que vous avez préalablement créé un environnement tpGBM en suivant les instructions du TP0.



Pour ce faire, et pour vous faciliter la tâche, vous devrez installer les librairies qui contiennent les modèles et les fonctions de traitement appropriées. Veuillez suivre les indications suivantes :

Vous devrez commencer par installer des nouveaux modules, en spécifiant les bonnes versions.

```
$ conda activate tpGBM
$ conda install pytorch torchvision pytorch-cuda=12.4 -c pytorch -c nvidia
$ pip install datasets
```

! Assurez-vous que vous pouvez exécuter la première cellule du notebook (qui importe les différentes librairies) afin de vérifier que l'installation s'est déroulée correctement. Demandez de l'aide à vos chargés de laboratoire si l'exécution contient des erreurs.

i Instructions pour l'utilisation du notebook via Google Colab :

1. Ouvrez Google Colab. Vous aurez besoin d'un compte Google.
2. Cliquez sur **Open Colab**.
3. Une fenêtre s'ouvrira. Sélectionnez **Importer**, puis venez charger le notebook **tp2_seance2_colab.ipynb**.
4. En haut à droite, cliquez sur **Connecter**, puis sur la flèche vers le bas qui viendra d'apparaître, et enfin sur **Modifier le type d'exécution**. Choisissez l'option T4 GPU, puis **Enregistrer**. Vous devriez maintenant être connectés à un environnement avec GPU.
5. Exécutez la première cellule pour installer les librairies nécessaires au TP.
6. Cliquez sur l'icône  (**Files**) dans le panneau à gauche de l'écran pour accéder aux fichiers disponibles dans le notebook Colab. Cliquez ensuite sur l'icône  (**Upload to session storage**) pour charger le fichier **pcam_utils.py** qui vous est fourni.
7. Exécutez la deuxième cellule (qui importe les différentes librairies). Si tout fonctionne, vous pouvez maintenant faire le TP ! Si vous avez des problèmes, demandez de l'aide à vos chargés.

Attention ! Dans Colab le contenu du notebook comme tel (code dans les cellules, figures générées) sera sauvegardé automatiquement. Par contre, tous les *fichiers* seront perdus entre deux sessions. Vous devrez charger à chaque nouvelle session le fichier **pcam_utils.py**. Si vous avez sauvegardé un fichier (e.g. les poids d'un modèle), pensez à le télécharger localement sur votre ordinateur, puis le recharger dans Colab pour pouvoir le réutiliser plus tard.

Exercice III : Classification d'images histologiques

Le deep learning a révolutionné de nombreux domaines, et l'imagerie médicale ne fait pas exception. Aujourd'hui, les réseaux de neurones convolutifs (CNN) sont devenus omniprésents dans la détection, la classification et la segmentation des images médicales, offrant des performances inégalées dans l'analyse des données complexes. Dans ce TP, vous serez amenés à appliquer les concepts du deep learning au dataset PatchCamelyon (PCam), un ensemble de données dérivé de biopsies mammaires [1, 2]. PCam contient quelques centaines de milliers de petites images de tissus ($96 \text{ px} \times 96 \text{ px}$) annotées pour la présence ou l'absence de tissu cancéreux dans la région centrale de $32 \text{ px} \times 32 \text{ px}$. Ce TP vise à vous familiariser avec l'entraînement d'un modèle de classification binaire capable de reconnaître des tissus cancéreux, un exemple emblématique de l'application du deep learning en pathologie numérique.

Exploration des données

Q1. Combien d'images y a-t-il dans chaque sous-ensemble du dataset ?

! Attention! Si vous essayez de charger toutes les images d'un coup, ça ne marchera probablement pas car il y en a beaucoup trop pour la quantité de RAM qui est vraisemblablement disponible sur votre machine.

Q2. Quelles sont les classes représentées dans le dataset ? Comment sont-elles représentées ?

i La fonction `np.unique` pourrait vous être utile.

Q3. Quel nombre et proportion des images de chaque sous-ensemble appartient aux différentes classes ? Affichez un histogramme de la distribution des classes pour chaque sous-ensemble.

i Vous pouvez utiliser l'argument `edgecolor="black"` (e.g. `plt.hist(..., edgecolor="black")` ou `'ax.hist(..., edgecolor="black")`) pour tracer une ligne noire autour des bandes de votre histogramme. Ceci pourrait rendre la lecture des histogrammes plus facile.

Q4. Affichez un échantillon de l'ensemble d'entraînement. Assurez vous que votre échantillon contienne au moins 2 images de chaque classe. Affichez l'indice et la classe de chaque image.

(Sur)-entraînement d'un modèle

Une approche fréquemment recommandée pour valider qu'une architecture donnée est correctement implémentée et qu'elle est effectivement capable d'apprendre quelque chose est d'utiliser un tout petit sous-ensemble des données d'entraînement et de volontairement sur-entraîner (*overfitting*) jusqu'à convergence sur ce sous-ensemble.

i Une librairie très souvent utilisée dans l'écosystème Python est **tqdm**. Elle permet de facilement créer des barres de progression pour le suivi d'une tâche itérative. Pour l'utiliser, il suffit d'encapsuler un objet itérable (e.g. un **range**, une liste, un dataloader, etc.) avec la syntaxe **for ... in tqdm(iterable): ...**. Les fonctions fournies en font déjà usage pour vous donner un feedback visuel lors de l'entraînement. N'hésitez pas à vous en servir pour votre propre code pour faciliter le suivi de vos expériences (e.g. **for i in tqdm(range(num_epochs)):** ...).

Q5. Créez un modèle qui n'a pas été pré-entraîné (utilisez les valeurs par défaut de **pretrained** et **lr**). Sur-entraînez le sur 1% des données d'entraînement (50 époques devraient suffire). Affichez la courbe d'évolution de votre erreur (*loss*) d'entraînement. Évaluez ensuite sur le même sous-ensemble (i.e. avec le même dataloader) et affichez la *loss* et l'*accuracy* finales.

N.B. : Pour cette question, vous n'avez pas besoin de faire une validation à chaque époque. Faites uniquement l'entraînement du modèle puis évaluez-le ensuite *sur l'ensemble d'entraînement*.

Q6. Que pouvez-vous conclure sur les capacités d'apprentissage de votre modèle ?

Q7. Quelles sont les performances *sur l'ensemble de validation* de ce modèle sur-entraîné ?

Variation du taux d'apprentissage

! Attention ! Pour la suite du TP on veut utiliser le plus d'images possible du dataset, mais en faisant un compromis avec le temps d'entraînement. Le temps que vous écriviez votre code et que vous le testiez, vous pouvez utiliser une petite proportion des données. Toutefois pour le rendu final, utilisez **un minimum de 25%** des données d'entraînement. Vous devrez créer de nouveaux dataloaders avec la proportion appropriée.

Un des paramètres essentiels dans l'entraînement d'un modèle est le taux d'apprentissage (*learning rate*), qui détermine la taille du pas qu'on prend lors de la descente de gradient pour optimiser un modèle.

Q8. Entraînez 3 modèles avec respectivement un *learning rate* de **1**, **1e-3**, et **1e-6** pendant au moins 10 époques. Assurez-vous de laisser l'argument **pretrained** de la fonction **get_model_and_optimizer** à sa valeur par défaut. Durant l'entraînement de chaque modèle, faites une sauvegarde du modèle lorsque vous obtenez la meilleure accuracy. Affichez les courbes d'évolution de votre *loss* d'entraînement (par batch), et de votre *loss* et *accuracy* de validation (par époque).

? Pour sauvegarder un modèle, vous pouvez utiliser la syntaxe suivante :

```
torch.save(model.state_dict(), f"mon_modèle.pt")
```

La méthode **state_dict()** d'un modèle PyTorch retourne un **dict** qui contient les poids du modèle, tandis que la fonction **torch.save** permet de sauvegarder un tenseur (ou une collection de tenseurs comme un **dict**) sous un nom de fichier donné.

Pour charger un modèle à partir d'une sauvegarde, il faut d'abord créer un modèle identique, charger le **dict** des poids depuis le fichier, puis charger le **dict** de poids dans le modèle.

```
saved_model = get_model_and_optimizer(pretrained=False)[0]
```

```
saved_state_dict = torch.load("mon_modèle.pt", map_location="cpu")
```

```
saved_model.load_state_dict(saved_state_dict)
```

Si vous travaillez sur Google Colab, pensez à télécharger vos modèles sauvegardés, parce qu'ils ne persisteront pas d'une session à l'autre. Vous pourrez les recharger pour les réutiliser.

Q9. Que pouvez-vous conclure sur l'impact du taux d'apprentissage sur l'entraînement de

vosre modèle ? Quel est le meilleur taux d'apprentissage pour ce problème ?

Q10. Comparez votre meilleur modèle avec celui qu'on a sur-entraîné plus tôt avec le sous-ensemble de 1%. Comment expliquez-vous les différences de performance ?

Apprentissage par transfert

Une technique standard pour améliorer les performances d'un modèle est d'utiliser l'apprentissage par transfert (*transfer learning*) où on utilise un modèle qui a préalablement été entraîné sur une grande quantité de données à résoudre une tâche connexe à la nôtre. On procède à un affinage (*finetuning*) de ce modèle sur des nouvelles données pour le spécialiser sur une nouvelle tâche.

i Pour faire de l'apprentissage par transfert, on peut techniquement affiner n'importe quel modèle pré-existant. Un choix extrêmement commun comme point de départ est d'utiliser un modèle pré-entraîné sur le dataset ImageNet.

ImageNet est une vaste base de données d'images étiquetées. Elle contient des millions d'images classées en milliers de catégories, allant des objets du quotidien aux animaux. ImageNet a joué un rôle central dans les progrès du deep learning, notamment avec le défi ImageNet Large Scale Visual Recognition Challenge (ILSVRC), qui a permis le développement de réseaux neuronaux profonds.

En apprentissage par transfert, les modèles pré-entraînés sur ImageNet sont souvent utilisés comme point de départ, car ils ont déjà appris des caractéristiques visuelles générales à partir d'un large éventail d'images.

Q11. Étant données les différences marquées entre les caractéristiques des images d'ImageNet (des chats, des chiens, des autobus, des spatules, etc.) et les images histologiques (des cellules colorées), pensez-vous qu'utiliser un modèle pré-entraîné sur ImageNet va aider vos performances de classification ? Expliquez votre raisonnement.

Q12. Affinez pendant au moins 10 époques un modèle pré-entraîné. Créez le en utilisant l'argument `pretrained=True`. Utilisez le *learning rate* qui vous a donné les meilleures performances à l'exercice précédent. Durant l'entraînement du modèle, faites une sauvegarde du modèle lorsque vous obtenez la meilleure *accuracy*. Affichez les courbes d'évolution de votre *loss* d'entraînement, et de votre *loss* et *accuracy* de validation.

Q13. Comparez les performances de votre modèle pré-entraîné avec celles de votre modèle non pré-entraîné. Qu'en concluez-vous ?

Évaluation finale

Jusqu'à présent on n'a évalué le modèle que sur le sous-ensemble de validation. Pour obtenir une évaluation finale de notre modèle, on doit l'évaluer sur l'ensemble de test.

Q14. Évaluez votre meilleur modèle sur l'ensemble de test. Affichez la *loss* et l'*accuracy* finale.

i Comme vous avez sauvegardé le meilleur modèle que vous avez obtenu à chaque expérience, ici serait l'occasion de charger le meilleur d'entre eux !

Q15. Commentez les performances de votre modèle sur l'ensemble de test et comparez-les à celles obtenues sur l'ensemble de validation.

Q16. Pourquoi est-il important de maintenir un ensemble de test séparé de l'ensemble de validation ? Que se passerait-il si on n'avait que deux ensembles (entraînement et validation/test) plutôt que trois (entraînement, validation, test) ?

Q17. Au sein de l'ensemble de test, identifiez

- Un exemple d'un vrai positif (une image cancéreuse correctement identifiée comme cancéreuse).
- Un exemple d'un faux positif (une image non-cancéreuse incorrectement identifiée comme cancéreuse).
- Un exemple d'un faux négatif (une image cancéreuse incorrectement identifiée comme non-cancéreuse).
- Un exemple d'un vrai négatif (une image non-cancéreuse correctement identifiée comme non-cancéreuse).

Affichez chaque image, son label, la prédiction de votre modèle, et l'index de l'exemple.

Q18. Au vu des performances de votre meilleur modèle sur les données de test, seriez-vous à l'aise d'intégrer ce modèle dans des protocoles cliniques d'histopathologie ?

Références

- [1] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen et M. Welling, "Rotation equivariant cnns for digital pathology," dans *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, 2018, p. 210–218.
- [2] B. E. Bejnordi, M. Veta, P. J. Van Diest, B. Van Ginneken, N. Karssemeijer, G. Litjens,

J. A. Van Der Laak, M. Hermsen, Q. F. Manson, M. Balkenhol *et al.*, “Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer,” *JAMA*, vol. 318, n°. 22, p. 2199–2210, 2017.