Simon
CHANTHRABOUTH-LIEBBE
ING 3 IA Groupe B

Rapport de mobilité internationale

NATIONAL
YANG MING CHIAO TUNG
UNIVERSITY

Inversion des spectres de dispersion par deep learning

Laboratoire de génie civil
NYCU Hsinchu, Taïwan

Du 15 juin au 28 août 2026

Supervision : Prof. Chih-Ping Lin
Encadrement du projet : Dr. Kinh Tran

Année scolaire : 2025 - 2026

Remerciements

Avant de présenter le travail réalisé pendant cette mobilité internationale, je tiens à remercier les personnes qui m'ont accompagné au cours de mon stage à Taïwan. Cette expérience représentait pour moi une découverte à plusieurs niveaux : celle d'un pays, d'un environnement de recherche et d'un domaine d'application de l'intelligence artificielle que je connaissais encore peu.

Je remercie en premier lieu le professeur Chih-Ping Lin, qui m'a accueilli au Civil Engineering Lab de National Yang Ming Chiao Tung University, à Hsinchu. En me permettant de rejoindre son équipe pendant deux mois et demi, il m'a donné l'occasion de mettre mes compétences en informatique au service d'un problème concret de génie civil. Cette confiance m'a permis de découvrir le fonctionnement d'un laboratoire et de participer à un projet de recherche déjà engagé.

Je souhaite également remercier Dr. Kinh Tran, qui menait le projet auquel j'ai contribué et qui m'a encadré au quotidien. Ses explications sur la propagation des ondes, les données et les attentes du projet m'ont aidé à comprendre le sens physique du travail que je réalisais. Les échanges que nous avons eus ont accompagné mes choix techniques, mais aussi ma manière d'interpréter les résultats et de définir les expériences suivantes.

Je remercie les membres du laboratoire pour leur accueil et pour les échanges lors des réunions. Présenter mon travail en anglais devant des personnes issues d'un autre domaine scientifique a été un exercice formateur. Il fallait rendre mes choix compréhensibles, expliquer ce que les modèles apportaient et être capable de discuter de leurs limites.

Enfin, je remercie mon école CY Tech pour la formation dans laquelle s'inscrit cette mobilité. Ce stage m'a permis de relier les connaissances acquises en spécialité intelligence artificielle à une application de recherche, tout en développant mon autonomie dans un contexte international.

Je vous remercie !

NATIONAL YANG MING CHIAO TUNG UNIVERSITY

CY TECH

Sommaire

Introduction 4
Présentation de l'organisme d'accueil 5
1) Présentation générale 5
2) Organisation et environnement de travail 5
3) Mes impressions sur le laboratoire 5
Problématique 6
Comprendre le sol sans le creuser 6
La difficulté du problème inverse 7
Ce que le modèle doit réellement prédire 7
Missions 8
Mission 1 : prendre en main et comparer les modèles 8
Mission 2 : améliorer la représentation des données 8
Organisation et livrables 8
État de l'Art 10
Des courbes de dispersion au spectre complet 10
L'apport de l'apprentissage profond 10
Le choix des architectures de réseaux 11
Géométrie et augmentation des données 11
Réalisation des missions 12
Phase 1 : comprendre le code et les données 12
Phase 2 : comparer les architectures 14
Phase 3 : tester l'ajout de bruit 16
Phase 4 : repenser la branche géométrique 17
Phase 5 : prolonger les essais et préparer la suite 20
Les difficultés rencontrées et la gestion du projet 22
Conclusion 24
Bilan des résultats 24
Compétences acquises 24
Bilan personnel et interculturel 25
Bibliographie 26
Annexes 28
Annexe 1 : repères et organisation 28
Annexe 2 : code et calculs de paramètres 29
Annexe 3 : résultats complémentaires 30
Annexe 4 : configuration et reprise du projet 31

Introduction

Ce rapport présente le stage de mobilité internationale que j'ai réalisé du 15 juin au 28 août 2026 au sein de National Yang Ming Chiao Tung University, à Hsinchu, à Taïwan. Étudiant en troisième année du cycle ingénieur à CY TECH, en spécialité intelligence artificielle, j'ai rejoint temporairement le laboratoire de génie civil sous la supervision du professeur Chih-Ping Lin et l'encadrement de Dr. Kinh Tran. C'était ma première expérience dans un laboratoire de recherche.

Pour comprendre l'intérêt de cette mission, il est utile de partir d'une question simple : comment savoir ce qui se trouve sous nos pieds sans creuser ? Dans le projet du laboratoire, la réponse passe par les ondes. Un impact est produit à la surface du sol à l'aide d'un marteau, puis une ligne de capteurs enregistre les vibrations qui se propagent. À partir de ces mesures, on cherche à retrouver les variations des propriétés mécaniques du sol et à mieux distinguer les couches qui le composent.

La difficulté est que les capteurs ne donnent pas directement l'épaisseur des couches ni leur vitesse de cisaillement. Ils mesurent une réponse en surface qu'il faut interpréter. Il s'agit donc d'un problème inverse : on observe les effets de la propagation et l'on essaie de retrouver les caractéristiques du milieu qui les a produits. Ma contribution consistait à utiliser l'apprentissage profond pour estimer un profil de vitesse des ondes de cisaillement, noté *Vs*, à partir d'un spectre de dispersion [1].

Le laboratoire disposait déjà d'un cadre de simulation, de données synthétiques et d'un premier réseau de neurones. Mon rôle était de comprendre cet environnement, de comparer plusieurs architectures et de proposer des améliorations. Le temps disponible, environ deux mois et demi, demandait de faire des choix : chaque entraînement pouvait prendre plusieurs heures à plusieurs jours, et toutes les idées ne pouvaient pas être testées dans les mêmes conditions.

Mon travail s'est progressivement organisé autour de trois axes : la comparaison des réseaux, l'ajout de bruit aux données et la refonte de la branche qui représente la géométrie des capteurs. Cette dernière partie a donné le résultat le plus marquant du stage. En remplaçant une représentation inutilement volumineuse par un encodage plus compact, j'ai obtenu des modèles beaucoup plus petits, avec une précision de validation très proche de celle des versions initiales.

Ce rapport présente l'organisme d'accueil, la problématique et les missions, avant de situer les méthodes utilisées dans leur contexte scientifique. Je détaille ensuite les étapes de réalisation, les résultats et les difficultés rencontrées, puis le bilan technique et personnel de cette expérience. Les essais à 300 000 données enregistrés au début du mois de septembre sont présentés comme un prolongement du travail, distinct de la période officielle du stage.

Présentation de l'organisme d'accueil

1) Présentation générale

National Yang Ming Chiao Tung University, généralement appelée NYCU, est issue de la fusion de National Yang Ming University et de National Chiao Tung University en 2021. J'ai été accueilli au Civil Engineering Lab, à Hsinchu, dans l'environnement du département de génie civil [2]. Les travaux du professeur Chih-Ping Lin concernent notamment la géophysique appliquée à l'ingénierie, la géotechnique, l'instrumentation et les problèmes inverses [3].

Le projet auquel j'ai participé relie ces domaines à l'intelligence artificielle. L'équipe apporte les connaissances physiques nécessaires pour produire et interpréter les données ; le deep learning est étudié pour estimer les propriétés du sol. Dans ce cadre, faire fonctionner le code ne suffit pas : il faut aussi expliquer ce qu'une expérience permet de vérifier et ce que son résultat signifie.

2) Organisation et environnement de travail

Le professeur Lin assurait la supervision scientifique de mon stage. Dr. Kinh Tran, qui menait le projet, était mon interlocuteur principal pour comprendre les données et choisir les pistes à privilégier. Les réunions hebdomadaires permettaient de présenter l'avancement, et les échanges en petit groupe d'approfondir les difficultés. J'ai présenté mon travail à mes encadrants le 21 août, puis au laboratoire le 25 août, en anglais.

Je développais sur mon ordinateur portable fourni par CY Tech, puis utilisais GitHub pour transférer le code vers la station du laboratoire, accessible à distance avec AnyDesk. Sa carte NVIDIA RTX 4080 SUPER permettait les entraînements avec PyTorch et CUDA. Pendant les calculs, je préparais les essais suivants, lisais des articles ou analysais les résultats précédents.

3) Mes impressions sur le laboratoire

Pour cette première expérience de recherche, j'ai apprécié la liberté d'organisation : je pouvais adapter mes horaires et travailler certains jours à distance. En retour, je devais suivre les calculs, conserver les résultats et préparer la suite avec autonomie. Ce fonctionnement m'a permis de découvrir une autre manière de travailler.

Le rythme de l'équipe différait aussi de mes repères en France. Certains membres restaient tard au laboratoire, parfois durant la nuit. Je retiens cette observation comme une caractéristique de mon environnement d'accueil. Elle m'a amené à réfléchir à ma propre organisation et à la façon de rester régulier dans un cadre aussi souple.

Problématique

Comprendre le sol sans le creuser

La connaissance du sol intervient notamment dans l'étude des fondations et le diagnostic des infrastructures. Les méthodes géophysiques permettent d'obtenir des informations à partir de mesures réalisées en surface. Dans notre cas, le dispositif comprend une source d'impact et plusieurs capteurs alignés, avec une distance fixe entre chaque capteur.

Trois paramètres décrivent cette géométrie : x0 représente la distance entre la source et le premier capteur, dx l'espacement entre les capteurs, et Ch le nombre de capteurs. La longueur totale de la ligne vaut donc L = (Ch − 1) × dx. Les signaux enregistrés sont ensuite traités pour obtenir un spectre de dispersion, qui représente l'énergie en fonction de la fréquence et de la vitesse de phase.

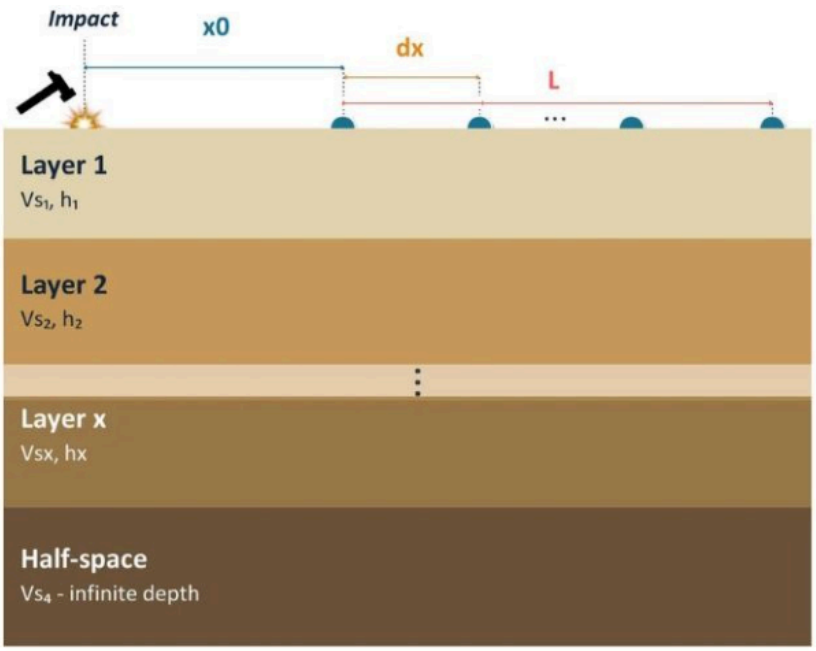


Figure 1 : Principe de l'acquisition MASW et paramètres géométriques. Schéma réalisé à partir de ma présentation et de Park et al. [1].

Cette méthode est appelée *MASW*, pour Multichannel Analysis of Surface Waves, ou analyse multicanal des ondes de surface. Elle utilise le fait que la propagation des ondes de surface dépend de la fréquence et des propriétés des couches traversées. En étudiant cette dispersion, on cherche à retrouver le profil de vitesse de cisaillement Vs en fonction de la profondeur [1].

Une distinction a été importante dans ma prise en main du sujet : la vitesse de phase visible dans le spectre et la vitesse de cisaillement recherchée ne sont pas la même grandeur. La première décrit la propagation d'un mode d'onde de surface ; la seconde caractérise le milieu. Le

réseau doit apprendre la relation entre ces informations, en tenant compte de la géométrie et de la structure du sol.

La difficulté du problème inverse

Un même spectre, ou des spectres très proches, peut correspondre à plusieurs profils plausibles lorsque la bande de fréquence est limitée. Le bruit et la résolution du dispositif compliquent encore cette interprétation. On ne peut donc pas simplement associer une fréquence à une couche précise ni considérer qu'une faible erreur sur les exemples connus suffit à résoudre le problème.

Le laboratoire génère des données synthétiques à partir de profils dont les propriétés sont connues. Le calcul direct produit le spectre à partir d'un modèle de sol et d'une géométrie ; le réseau apprend ensuite à effectuer le chemin inverse.

Ce que le modèle doit réellement prédire

L'objectif général du projet concerne les vitesses et les couches du sol. Dans la version Model_100 sur laquelle j'ai principalement travaillé, la sortie du réseau est un vecteur de 101 valeurs décrivant un profil Vs discrétisé. Il ne s'agit pas de 101 couches ni d'une liste de matériaux. Les changements de couche se lisent dans les variations du profil ; leur épaisseur demande ensuite d'interpréter les interfaces et l'axe des profondeurs.

Une autre représentation, appelée Model_6 dans mes notes, utilise des épaisseurs et des vitesses explicites. Elle a été une piste secondaire. Dans ce rapport, les comparaisons principales portent sur Model_100 afin de conserver une même définition de la sortie et de ne pas mélanger des expériences qui répondent à des formulations différentes.

La question centrale de mon stage était donc la suivante : comment rendre cette estimation du profil Vs plus efficace, tout en conservant une qualité de reconstruction satisfaisante ? J'ai retenu plusieurs critères pour y répondre : l'erreur de validation, l'écart avec l'entraînement, le nombre de paramètres, le temps de calcul observé et la forme des profils prédits. Il fallait également produire un travail que Dr. Kinh pourrait reprendre après mon départ.

Missions

Les missions qui m'ont été confiées s'inscrivaient dans un même projet de recherche. Elles ont évolué au fur et à mesure des résultats et des discussions avec mes encadrants, tout en conservant le même objectif : améliorer l'inversion des spectres de dispersion par apprentissage profond.

Mission 1 : prendre en main et comparer les modèles

La première étape consistait à comprendre les données et le code existant. Avant de modifier une architecture, je devais savoir ce que représentaient les canaux d'entrée, comment les valeurs étaient normalisées et comment le profil cible était construit. J'ai ensuite organisé les comparaisons entre le CNN initial (ResNet50 baseline), ResNet-50 révisé, ResNet-34, DenseNet-121, EfficientNet-B0 et Swin-T.

Enjeux et objectifs

- Comprendre les fichiers d'entrée et leur signification physique.
- Enregistrer les métriques et les paramètres de chaque entraînement.
- Comparer plusieurs familles de réseaux dans un cadre commun.
- Identifier les modèles les plus intéressants en termes de précision et de taille.

Mission 2 : améliorer la représentation des données

Une deuxième partie de la mission concernait la manière de présenter l'information au réseau. J'ai testé l'ajout de bruit gaussien sur l'amplitude du spectre, puis travaillé sur la branche géométrique. Cette dernière est devenue prioritaire en fin de stage, lorsque son coût en paramètres est apparu disproportionné par rapport à l'information qu'elle recevait.

Enjeux et objectifs

- Étudier l'effet d'une première augmentation des données par du bruit.
- Comprendre l'écart entre les erreurs d'entraînement et de validation.
- Remplacer la répétition d'un scalaire par une représentation plus compacte.
- Intégrer les caractéristiques du dispositif de mesure dans les modèles retenus.

Organisation et livrables

Le calendrier général de ces étapes est repris en annexe 1.

Un autre sujet de deep learning avait été évoqué au laboratoire. Compte tenu du temps disponible, j'ai choisi de concentrer mes efforts sur le projet encadré par Dr. Kinh. Ce choix m'a permis d'approfondir une même chaîne de traitement et d'aboutir à des comparaisons

exploitables. Les livrables comprennent le dépôt GitHub, les architectures adaptées, les modèles transmis à Dr. Kinh, les journaux d'entraînement et ma présentation de fin de stage.

État de l'Art

Pour comprendre les choix effectués pendant le stage, il est nécessaire de replacer le projet dans son contexte scientifique. L'inversion des ondes de surface est un problème étudié depuis longtemps en géophysique. L'apprentissage profond propose une autre manière de l'aborder, mais son intérêt dépend autant de la qualité des données et du protocole que du réseau choisi.

Des courbes de dispersion au spectre complet

Dans la démarche MASW classique, les enregistrements multicanaux servent à identifier les caractéristiques de dispersion, puis à les inverser pour estimer le sol [1]. Cette étape peut devenir délicate lorsque plusieurs modes sont présents ou proches les uns des autres. Exploiter directement le spectre complet permet de conserver davantage d'information et de réduire la dépendance à une sélection manuelle des courbes.

Les travaux de *Wu et al.* étudient une inversion de spectres par une méthode Monte-Carlo à chaînes de Markov. Cette approche s'appuie sur le calcul physique pour explorer plusieurs configurations de couches. Elle présente un intérêt pour examiner des solutions alternatives et leurs incertitudes, mais demande de nombreuses évaluations du modèle direct [4]. Dans le cadre de mon stage, elle constituait un point de comparaison utile avec une méthode dont l'objectif est de produire rapidement une estimation après l'apprentissage.

L'apport de l'apprentissage profond

Cho et al. proposent de prédire directement un profil Vs à partir d'images de dispersion, puis de régulariser la sortie avec un réseau de post-traitement. Leur étude montre l'intérêt de l'approche, tout en soulignant les difficultés rencontrées lorsque les exemples s'éloignent de la distribution d'apprentissage [5]. Cette limite est directement liée à notre projet : un modèle peut être précis sur les profils simulés et moins fiable sur une mesure réelle qui présente d'autres caractéristiques.

Les travaux de *Tran et al.*, menés dans le contexte scientifique du laboratoire, exploitent les spectres de dispersion pour retrouver des profils de chaussée. La génération des données repose sur une simulation du champ d'ondes, puis le réseau apprend à relier les spectres aux paramètres recherchés. Le coût est principalement concentré dans la préparation des données et l'entraînement ; une fois le modèle appris, la prédiction demande une évaluation du réseau. L'étude examine également la comparaison avec une inversion Monte-Carlo et la possibilité d'affiner la solution par optimisation [6].

Ces méthodes peuvent donc être complémentaires. Une inversion fondée sur le modèle physique explore les solutions possibles, tandis qu'un réseau fournit une estimation rapide à

partir de ce qu'il a appris. Pour ma mission, l'enjeu était d'améliorer cette estimation tout en gardant à l'esprit qu'elle reste dépendante des profils et des géométries présents dans les données.

Le choix des architectures de réseaux

Les spectres sont représentés sous forme de grilles, ce qui permet d'utiliser des architectures initialement développées pour les images. Elles n'exploitent cependant pas toutes l'information de la même manière. ResNet utilise des connexions résiduelles pour faciliter l'apprentissage de réseaux profonds [7]. DenseNet réutilise les caractéristiques par concaténation entre les couches [8]. EfficientNet associe des blocs convolutifs compacts à une stratégie de dimensionnement du réseau [9]. Swin Transformer adopte une autre approche, fondée sur l'attention dans des fenêtres locales décalées [10]. L'idée était intéressante pour étudier des relations dans le spectre au-delà des filtres convolutifs habituels. Il ne faut toutefois pas le décrire comme un modèle sans aucune convolution ni aucune localité : la projection initiale des patches utilise une convolution et l'attention reste organisée par fenêtres.

Dans mon travail, ces architectures n'étaient pas utilisées pour classer des images. J'ai adapté leur entrée aux données scientifiques et remplacé leur sortie par une tête de régression. Le préentraînement apporte un point de départ, mais certaines couches sont remplacées et doivent être apprises pour cette tâche. Comparer les réseaux revient donc à comparer des modèles adaptés au problème, avec les limites que cela implique.

Géométrie et augmentation des données

Le spectre dépend aussi du dispositif de mesure. Il est donc logique de donner au réseau des informations sur la position et l'espacement des capteurs. La méthode FiLM, présentée par *Perez et al.*, permet de moduler des caractéristiques à l'aide d'une information de conditionnement [11]. Dans notre cas, cette information est géométrique : elle sert à ajuster les caractéristiques extraites du spectre avant la prédiction du profil.

L'ajout de bruit répondait à une autre question : comment préparer un modèle entraîné sur des données synthétiques propres à des mesures plus perturbées ? Des travaux comme *DeepDenoiser* montrent l'intérêt de l'apprentissage profond pour traiter du bruit dans des signaux sismiques [12]. Leur tâche est différente de la nôtre, mais elle rappelle l'importance de la représentation du bruit. Un bruit gaussien ajouté à une image normalisée reste une première hypothèse ; il ne reproduit pas toutes les perturbations d'une acquisition réelle.

L'état de l'art m'a ainsi aidé à formuler les expériences sans considérer qu'une architecture récente serait automatiquement meilleure. Je devais vérifier ce que chaque modification apportait sur nos données, puis replacer le résultat dans les conditions précises de l'essai.

Réalisation des missions

Phase 1 : comprendre le code et les données

Pour comprendre le travail réalisé, il faut d'abord présenter l'organisation du projet. Le dépôt contient plusieurs dossiers correspondant à des représentations et à des expériences différentes. Mon travail principal porte sur Model_100. Les dossiers log et figures conservent les traces des entraînements et leurs représentations graphiques [S3].

Les cinq scripts centraux se répartissent les responsabilités de manière assez claire. **main.py** prépare la configuration et lance les opérations ; **model.py** contient le modèle initial et plusieurs variantes ; **model_test_geo.py** regroupe les architectures utilisant la géométrie enrichie ; **call_dataset.py** prépare les exemples ; **loss_fcns.py** définit les fonctions de coût. Cette séparation m'a permis de faire évoluer une partie du système sans perdre de vue le chemin complet des données.

Tableau 1 : Responsabilités des cinq scripts centraux de Model_100.

Fichier	Responsabilité principale
main.py	Configuration, construction des modèles et lancement train ou test
model.py	Modèle initial, backbones comparés et premières variantes géométriques
model_test_geo.py	Descripteurs géométriques, Fourier, FiLM de faible rang et modèles associés
Call_dataset.py	Lecture des .mat et .csv, préparation des tenseurs et augmentations
loss_fcns.py	Définition de RMSELoss et de fonctions de coût alternatives

La représentation du spectre

Le spectre fourni au réseau contient trois canaux sur une grille de 76 lignes et de 191 colonnes. Ces canaux ne correspondent pas aux couleurs rouge, verte et bleue d'une photographie : ils décrivent les coordonnées normalisées de fréquence et de vitesse de phase, ainsi que l'amplitude. Les fréquences vont de 5 à 80 Hz avec un pas de 1 Hz, et les vitesses de phase de 50 à 1 000 m/s avec un pas de 5 m/s.

Tableau 2 : Structure du jeu de données décrit dans la présentation

Élément	Dimensions ou valeurs	Rôle
Spectre	3 × 76 × 191	Coordonnées normalisées et amplitude
Fréquence	5 à 80 Hz, pas de 1 Hz	76 lignes de fréquence
Vitesse de phase	50 à 1 000 m/s, pas de 5 m/s	191 colonnes
Géométrie	x0 de 2 à 20 m ; dx = 1, 1,5 ou 2 m ; Ch = 24 ou 48	Description de l'acquisition

Les coordonnées sont normalisées en tenant compte de la longueur L du dispositif et de la vitesse de référence Vsmax = 1 000 m/s. Le code utilise notamment f′ = fL / Vsmax, c′ = c / Vsmax et x0′ = x0 / L. Cette étape relie la représentation numérique aux échelles physiques de l'acquisition. Elle a été l'un des points que j'ai dû clarifier avec Dr. Kinh avant de pouvoir interpréter correctement les entrées.

L'amplitude est normalisée par un maximum calculé sur chaque ligne. Le bruit gaussien, lorsqu'il est activé, est ajouté à cette amplitude. Les valeurs négatives sont ramenées à zéro, mais les valeurs supérieures à un ne sont pas écrêtées dans cette opération. Le chargeur actuel mérite également une protection lorsqu'un maximum de ligne est nul, afin d'éviter une division invalide. Ces détails sont importants pour reprendre le code et conserver le même sens aux données.

Définir un protocole de comparaison

J'ai consacré une partie du début du stage à rendre les expériences plus faciles à suivre : enregistrer les résultats dans des CSV, fixer les graines pseudo-aléatoires et utiliser un arrêt anticipé. Cette instrumentation était nécessaire, car une courbe affichée pendant un entraînement ne suffit pas à comparer plusieurs essais réalisés à des moments différents.

La campagne initiale porte sur 90 000 données, avec une séparation annoncée de 80 % pour l'entraînement et de 20 % pour la validation, soit respectivement 72 000 et 18 000 exemples. Le protocole présenté au laboratoire utilise des batchs de 8, une graine de 42, l'optimiseur Adam avec un taux initial de 10^{-4}, puis une division de ce taux par deux toutes les 20 époch. Les entraînements vont jusqu'à 200 époch, avec une patience de 45 pour l'arrêt anticipé.

Fixer une graine rend les essais plus faciles à reproduire, mais ne donne pas des poids initiaux identiques à des architectures différentes. De même, les paramètres actuels de main.py ont évolué depuis cette première campagne. Je distingue donc le protocole des essais présentés et la configuration la plus récente du code, détaillée en annexe.

Mesurer l'erreur du modèle

La fonction RMSELoss calcule la racine de l'erreur quadratique moyenne entre les valeurs prédites et les valeurs cibles. Pour un batch de B profils contenant chacun 101 valeurs, son principe s'écrit :

$$RMSE = \sqrt{[\,(1 / (B \times 101)) \times \Sigma_i \, \Sigma\text{⍰} \, (\hat{y}_i\text{⍰} - y_i\text{⍰})^2\,]}$$

Une erreur faible indique que la prédiction est proche de la cible selon cette mesure. Elle ne dit toutefois pas, à elle seule, si une interface est bien localisée. J'ai donc utilisé les profils tracés en complément des métriques. Les valeurs étant calculées sur des sorties normalisées, je les conserve dans l'échelle des journaux au lieu de les convertir directement en m/s.

Le fichier loss_fcns.py contient aussi des fonctions alternatives, notamment liées aux gradients ou à des écarts relatifs. Leur présence dans le code ne signifie pas qu'elles ont toutes été utilisées dans les expériences présentées. Pour le bilan, je m'appuie sur les campagnes identifiées comme RMSELoss et sur leurs colonnes enregistrées. Les moyennes de RMSE par batch, la RMSE globale et la moyenne des RMSE par profil restent des indicateurs différents.

Phase 2 : comparer les architectures

La première série d'expériences avait pour objectif de déterminer si un autre extracteur de caractéristiques pouvait améliorer le compromis entre précision et taille. Le modèle initial est déjà fondé sur un ResNet-50, mais sa tête et sa branche géométrique sont volumineuses. J'ai comparé cette référence à plusieurs variantes en conservant le même type d'entrée et la même sortie de 101 valeurs.

Tableau 3 : Campagne initiale 90K sans bruit. M : million de paramètres.

Modèle	Paramètres M	épochs	Train	Validation
CNN initial	58,26	165	0,006657	0,063701
ResNet-50 révisé	57,28	88	0,007672	0,062664
ResNet-34	53,49	132	0,009425	0,064907
DenseNet-121	39,68	99	0,015927	0,060956
Swin-T	59,99	136	0,027425	0,066847
EfficientNet-B0	37,00	169	0,023875	0,062671

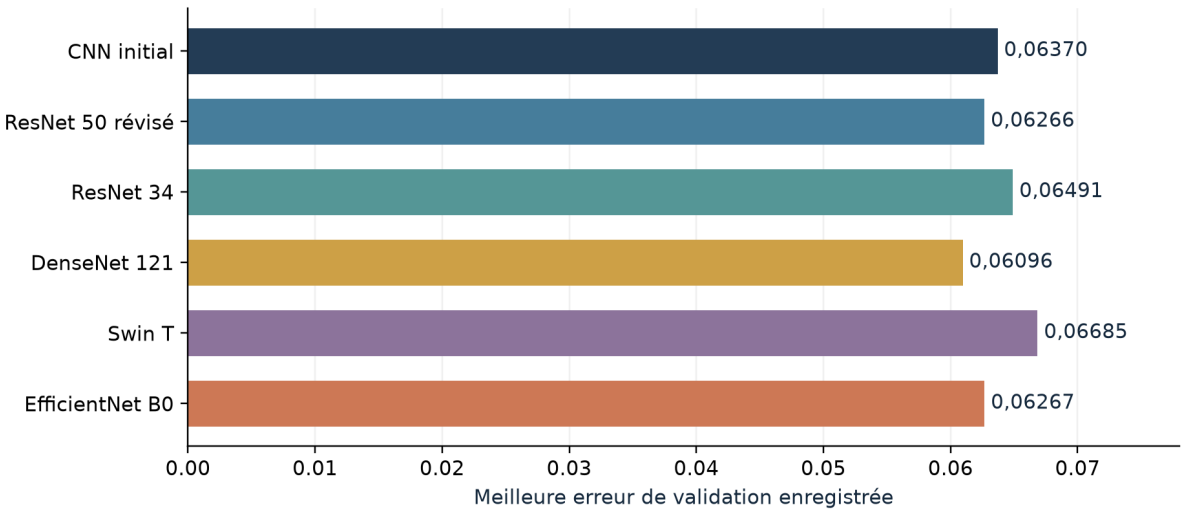


Figure 2 : Comparaison des architectures sur la campagne initiale à 90 000 données. Graphique calculé à partir des journaux d'entraînement CSV.

Dans cette campagne, DenseNet-121 obtient la plus faible erreur de validation parmi les six architectures, avec 0,060956. EfficientNet-B0 atteint 0,062671, pour environ 37 millions de paramètres contre 58,26 millions pour le CNN initial. Ce résultat m'a orienté vers ces deux

architectures pour la suite : DenseNet était intéressant pour la précision observée, et EfficientNetB0 pour sa taille plus faible.

Le remplacement du backbone ne résume toutefois pas toutes les différences entre les variantes. La fusion du modèle initial utilise une couche linéaire seule, tandis que certaines versions révisées y ajoutent une activation. Les comparaisons portent donc sur les modèles tels qu'ils ont été construits. Avec une seule graine principale, les petits écarts de précision doivent également être confirmés avant d'être considérés comme des gains stables.

Comprendre l'écart entre entraînement et validation

L'une des observations qui a guidé mon travail est l'écart entre la courbe d'entraînement et celle de validation. Sur le CNN initial, l'erreur d'entraînement continue de diminuer fortement, alors que la validation se stabilise. Le modèle apprend donc très bien les exemples qui lui sont présentés, sans retrouver le même niveau de précision sur les exemples réservés à la validation.

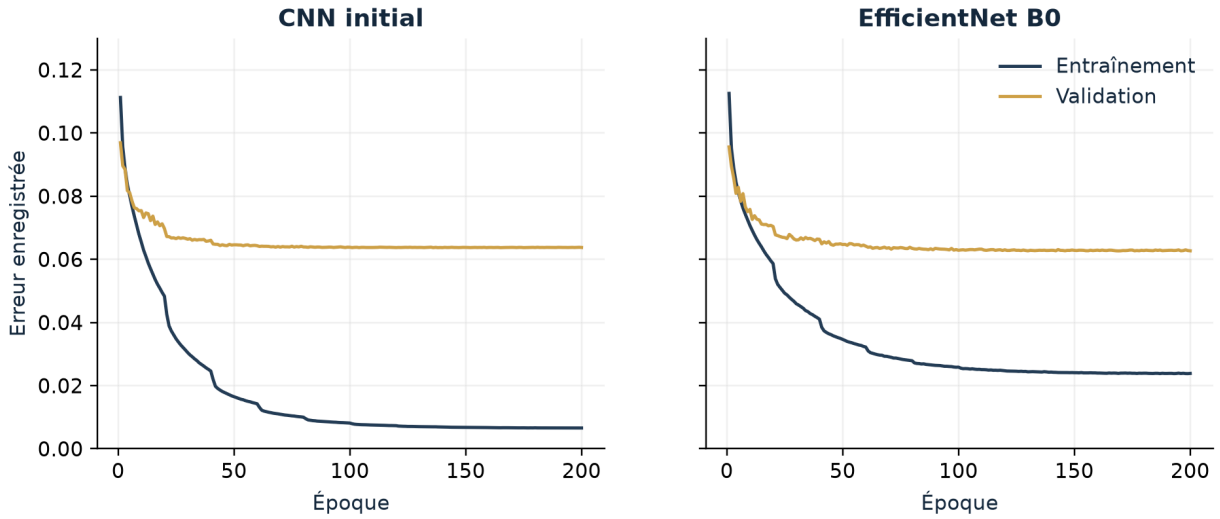


Figure 3 : Courbes du CNN initial et d'EfficientNet-B0 sans bruit. Graphiques tracés à partir des journaux avec 90K données.

À la meilleure époch du CNN initial, l'erreur d'entraînement est de 0,006657 et celle de validation de 0,063701, soit un rapport d'environ 9,57. Pour EfficientNet-B0, ce rapport est d'environ 2,62. Ces écarts sont compatibles avec du surapprentissage, même s'ils peuvent aussi être influencés par la distribution des données et le comportement des couches de normalisation.

Cette analyse m'a appris à ne pas juger un réseau uniquement sur la proximité de ses deux courbes. Swin-T présente un rapport encore plus faible, environ 2,44, mais son erreur de validation est la plus élevée des six modèles. Réduire l'écart entre entraînement et validation n'est donc intéressant que si la précision reste satisfaisante. Les fluctuations observées sur DenseNet ont également ouvert une réflexion sur la taille des batchs et la normalisation, sans expérience permettant d'en isoler la cause.

Phase 3 : tester l'ajout de bruit

L'ajout de bruit est venu d'une interrogation sur nos données : les exemples synthétiques sont très propres, alors que les acquisitions réelles peuvent être perturbées. J'ai donc testé un bruit gaussien appliqué à l'amplitude du spectre, avec des écarts types de 0,02, 0,05, 0,08 et 0,10. Ces valeurs se rapportent à l'amplitude normalisée ; elles ne représentent pas directement un pourcentage de bruit physique ni un rapport signal sur bruit mesuré sur le terrain.

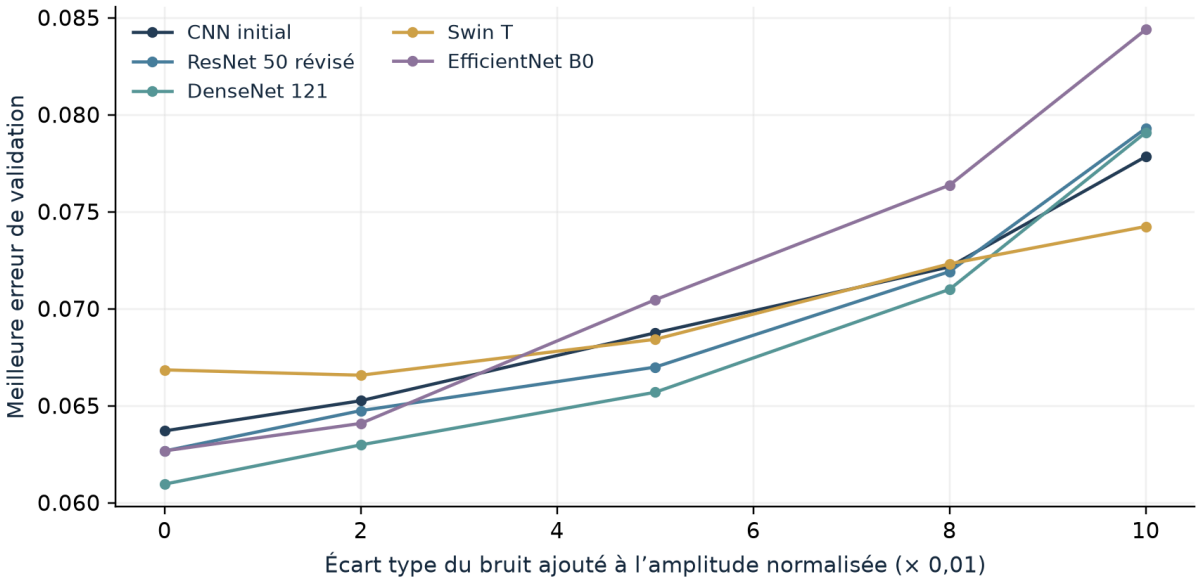


Figure 4 : Effet du bruit gaussien sur la meilleure erreur de validation. Valeurs issues des campagnes à 90K données, avec et sans bruit. Le tableau complet est donné en annexe.

Les résultats n'ont pas montré le bénéfice général que cette piste pouvait laisser espérer. Pour EfficientNet-B0, l'erreur passe de 0,062671 sans bruit à 0,064087 avec un écart type de 0,02, puis atteint 0,084391 à 0,10. DenseNet-121 suit la même tendance globale, de 0,060956 à 0,079087. Plus le bruit ajouté est fort, plus la précision sur la validation propre se dégrade dans ces essais.

Il existe une petite exception avec Swin-T : l'erreur passe de 0,066847 à 0,066571 avec le bruit de 0,02, soit une baisse d'environ 0,4 %. Sans répétitions indépendantes, cet écart reste trop faible pour conclure à un avantage robuste. Je retiens donc que le bruit gaussien n'a pas apporté d'amélioration générale dans le protocole étudié.

Cette étape a tout de même été utile. Elle m'a amené à distinguer deux questions : être précis sur des spectres propres et résister à des perturbations réalistes. Notre validation mesure surtout la première. Pour étudier la seconde, il faudrait définir des essais avec différents types de bruit et des mesures réelles disposant d'une référence. Le résultat ne signifie donc pas que toute augmentation est inutile ; il invite à mieux choisir les perturbations que l'on souhaite représenter.

Phase 4 : repenser la branche géométrique

Le changement le plus important est intervenu lorsque j'ai examiné la manière dont le modèle traitait la géométrie. La version initiale reçoit x0′, une seule valeur décrivant le déport normalisé de la source. Cette valeur est répétée sur une grille de 76 × 191 cellules, puis la grille est aplatie pour former un vecteur de 14 516 valeurs identiques. Ce vecteur traverse ensuite des couches de dimensions 14 516 → 2 048 → 512 → 128.

Cette branche contient à elle seule 30 845 568 paramètres, soit environ 52,9 % des paramètres du CNN initial. Autrement dit, plus de la moitié du modèle sert à traiter une information scalaire répétée. Le calcul de la première couche permet de comprendre pourquoi cette représentation est redondante :

$$y_i = \Sigma W_i x0' + b_i = x0' \times (\Sigma W_i) + b_i$$

Comme toutes les entrées sont identiques, leur contribution peut être représentée par un seul poids effectif correspondant à la somme des poids de la ligne. La répétition ne crée donc aucune information spatiale supplémentaire. Cette observation a été déterminante dans la discussion du 21 août : avant de chercher un réseau d'image plus complexe, il était pertinent de supprimer ce coût inutile.

Une première simplification avec un MLP

J'ai d'abord étudié une représentation directe du scalaire avec un petit perceptron multicouche, ou MLP. La variante ModelCNN_fvs_v2 applique une normalisation de batch puis les couches 1 → 32 → 128. Cette branche ne compte plus que 4 290 paramètres entraînables, et le modèle complet passe de 58 260 749 à 27 419 471 paramètres.

L'erreur minimale de validation passe de 0,063701 à 0,064400, soit une hausse relative d'environ 1,1 %. La précision reste donc proche pour une réduction de 52,9 % du nombre total de paramètres. Cette première expérience confirmait qu'une représentation beaucoup plus compacte pouvait suffire pour traiter le scalaire.

Cette simplification change aussi la manière dont l'optimisation est paramétrée. L'équivalence algébrique explique la redondance de la première couche, mais elle ne permet pas de déduire directement un facteur exact sur le taux d'apprentissage effectif d'Adam. Pour interpréter cette étape, je retiens surtout le gain mesurable en paramètres et le comportement observé en validation.

Enrichir la géométrie sans retrouver le même coût

La suite a consisté à intégrer davantage d'informations sur le dispositif. La version de model_test_geo.py utilise x0′, dx et Ch pour calculer huit descripteurs continus : ils comprennent notamment la longueur de la ligne, le déport physique, la distance du dernier capteur et des

rapports géométriques. Cinq variables catégorielles représentent les trois espacements possibles et les deux nombres de capteurs.

L'encodage est complété par quatre bandes de Fourier : chaque descripteur continu est associé à quatre sinus et quatre cosinus. On obtient ainsi 8 × 9 + 5 = 77 entrées, traitées par un MLP de dimensions 77 → 64 → 128. Cet encodeur compte 13 312 paramètres. Il fournit une représentation plus riche, dont les composants devront être comparés séparément pour mesurer précisément leur apport.

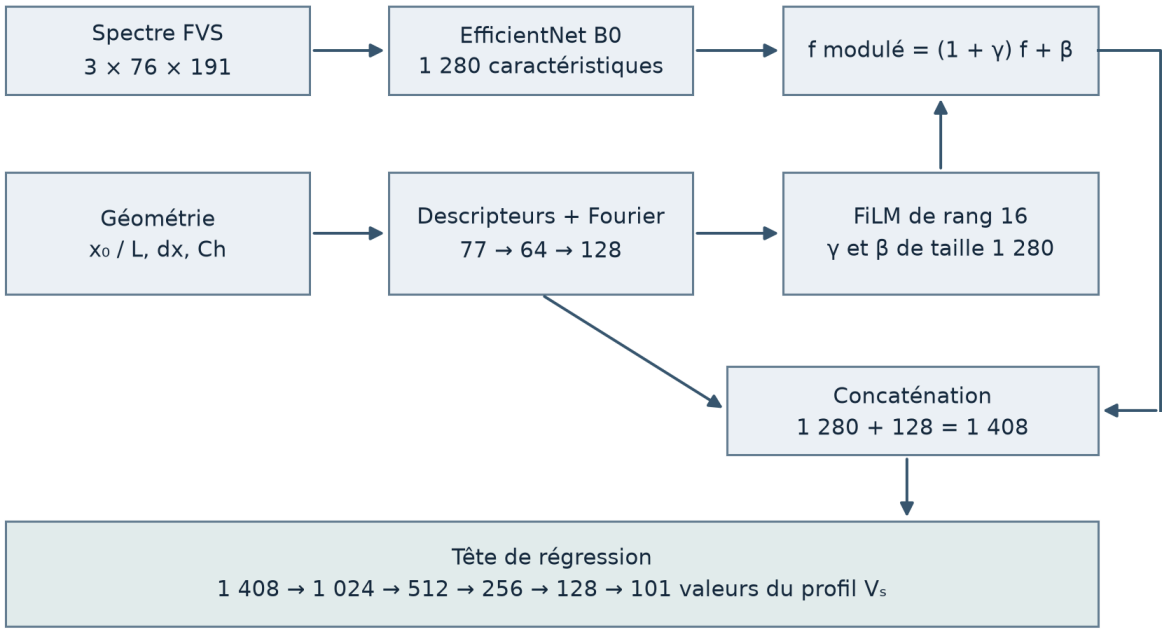


Figure 5 : Architecture d'EfficientNet-B0 avec la branche géométrique enrichie. Schéma réalisé d'après ModelEfficientNetB0_fvs_geo dans model_test_geo.py, pour les expériences à trois canaux.

La géométrie sert ensuite à moduler les 1 280 caractéristiques extraites du spectre par EfficientNet-B0. FiLM applique une transformation de la forme $\tilde{f} = (1 + \gamma) \odot f + \beta$, où γ et β dépendent de l'encodeur géométrique. Un générateur de rang 16 limite le nombre de paramètres. Sa sortie est initialisée à zéro pour que le modèle commence avec une modulation équivalente à l'identité.

Les caractéristiques spectrales modulées sont concaténées aux 128 caractéristiques géométriques, puis la tête de régression produit les 101 valeurs du profil. L'encodeur et FiLM totalisent 58 896 paramètres, soit environ 0,95 % du modèle final. Ce total est supérieur aux quelque 5 000 paramètres évoqués pour une variante précoce, car il comprend aussi l'enrichissement géométrique et la modulation.

Les résultats de cette refonte

Tableau 4 : Résultats des principales variantes géométriques.

Variante à 90K	Paramètres M	Minimum val_loss
CNN initial	58,26	0,063701
CNN scalaire V2	27,42	0,064400
CNN géométrie G1	27,42	0,064219
CNN géométrie G3	27,46	0,063030
DenseNet-121 + géométrie	8,89	0,060728
EfficientNet-B0 + géométrie	6,21	0,062505

Pour EfficientNet-B0, le modèle passe de 36 997 985 à 6 211 313 paramètres, soit une réduction de 83,2 %. Dans le même temps, l'erreur minimale de validation passe de 0,062671 à 0,062505. DenseNet-121 passe de 39 682 149 à 8 886 773 paramètres, soit une réduction de 77,6 %, avec une erreur de validation de 0,060728 contre 0,060956 auparavant.

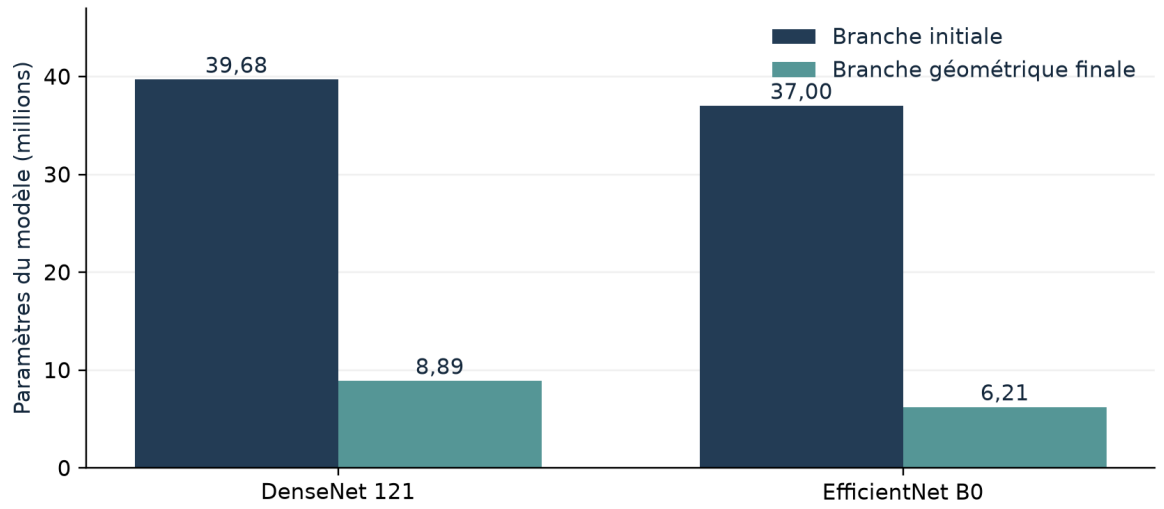


Figure 6 : Nombre total de paramètres avant et après la refonte de la branche géométrique. Valeurs num_params des journaux de la campagne 90K.

Ce résultat est celui que je retiens principalement de mon stage. Le gain ne réside pas dans une forte augmentation de la précision, puisque les différences d'erreur sont très petites. Il réside dans la possibilité de conserver cette précision avec des modèles beaucoup moins volumineux. Cette étape m'a montré l'intérêt de regarder la manière dont l'information est représentée, avant de modifier seulement le réseau qui traite l'image.

Le temps d'entraînement enregistré pour EfficientNet-B0 diminue également : la somme des durées d'époch passe d'environ 16,92 à 13,52 heures pour deux exécutions de 200 époch, soit environ 20 % de moins. Ce chiffre reste indicatif, car les conditions matérielles d'exécution ne sont pas entièrement contrôlées. Il mesure l'entraînement et ne doit pas être confondu avec une

mesure de rapidité en prédiction. Pour DenseNet, le nombre d'époch diffère entre les essais, ce qui limite la comparaison des durées totales.

Phase 5 : prolonger les essais et préparer la suite

Le passage à 300 000 données

Mes notes de fin de stage mentionnent le passage à 300K données. Le journal correspondant contient deux exécutions, DenseNet-121 et EfficientNet-B0 avec géométrie, datées des 1er et 2 septembre 2026. Elles sont donc présentées ici comme un prolongement expérimental après la fin officielle de ma mobilité. Les batchs passent à 64 exemples, avec une graine de 42.

Tableau 5 : Campagne 300K : métriques à l'époch du minimum de val_loss.

Modèle + géométrie	époch	val_loss	RMSE globale	RMSE moyenne par profil
DenseNet-121	134	0,012135	0,014622	0,004672
EfficientNet-B0	147	0,014227	0,016261	0,007339

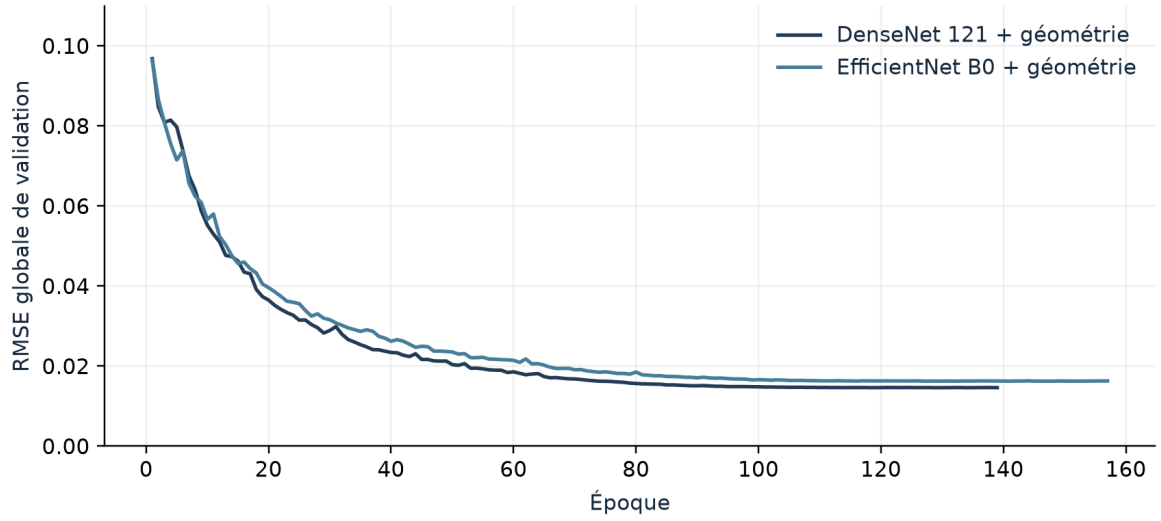


Figure 7 : RMSE globale de validation sur les expériences à 300 000 exemples. Courbes issues de val_rmse_pooled dans log_300k_v1.csv.

DenseNet atteint une erreur plus faible dans cette campagne, tandis qu'EfficientNet conserve l'avantage du nombre de paramètres. Les journaux contiennent respectivement 139 et 157 époch, pour environ 16,87 et 18,67 heures cumulées. Un modèle plus petit n'est donc pas nécessairement celui dont l'exécution complète est la plus courte.

Les valeurs du tableau sont relevées à l'époch où val_loss est minimale pour chaque modèle. Elles ne correspondent pas forcément au minimum de toutes les autres métriques. Les dernières lignes marquées is_best se situent aux époch 114 pour DenseNet et 132 pour

EfficientNet, alors que les minima bruts de val_loss arrivent aux époch 134 et 147. Cette distinction est utile pour retrouver le point de sauvegarde effectivement retenu.

La baisse des erreurs par rapport à la campagne 90K est importante, mais plusieurs éléments ont changé en même temps : le volume de données, la taille des batchs et le pibatchage de l'apprentissage. Il faudrait reprendre un protocole identique et un jeu de test commun pour attribuer précisément le gain à l'un de ces facteurs. Je conserve donc cette campagne comme un résultat complémentaire, sans l'assimiler à une comparaison où seul le nombre d'exemples aurait changé.

Observer les profils reconstruits

Les courbes d'erreur doivent être complétées par l'observation des profils. La figure suivante, reprise de ma présentation, montre un exemple pour DenseNet-121. Au début de l'apprentissage, la prédiction est lissée et suit mal les ruptures de la cible. À la meilleure époch, elle se rapproche davantage des changements de vitesse et des interfaces.

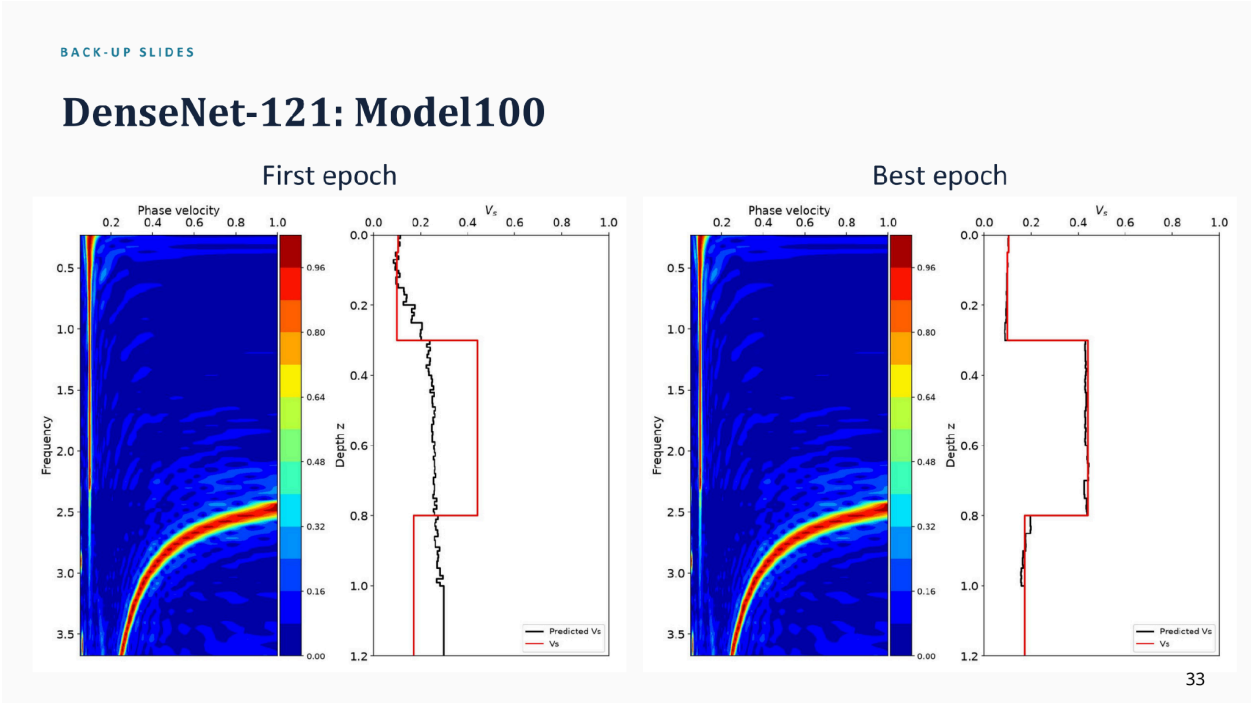


Figure 8 : Exemple de reconstruction avec DenseNet-121 et Model_100, à la première et à la meilleure époch. Reproduction de la diapositive 33 de ma présentation [S1]. Les axes sont normalisés ; il s'agit d'un exemple, et non d'une statistique sur le jeu de test.

Cet exemple rejoint l'objectif initial de rendre les transitions entre couches plus lisibles. Il ne permet toutefois pas de connaître la fréquence des erreurs sur tout le jeu de données. Une analyse plus complète devrait examiner les couches fines, les inversions de vitesse, l'erreur selon la profondeur et les cas où la bande de fréquence apporte peu d'information.

La question des mesures réelles reste également ouverte. Mes notes évoquent un essai envisagé sur une acquisition dont le profil de référence était inconnu, et le dépôt contient un dossier testing_measured. Une prédiction peut alors être examinée pour sa plausibilité, mais il manque une référence indépendante pour mesurer son exactitude. Les résultats présentés portent donc principalement sur des données synthétiques ; ils ne permettent pas d'affirmer que toutes les couches d'un terrain réel seraient correctement identifiées.

Le masquage des bandes de fréquence

Une autre piste discutée en fin de stage concernait les spectres incomplets. L'idée était d'apprendre au réseau à travailler lorsque certaines fréquences ne sont pas disponibles. Ma présentation indiquait ce sujet comme un travail en cours. Dans le code actuel, le masquage est devenu une option de Call_dataset.py et de main.py, mais les journaux historiques ne permettent pas de lui attribuer les résultats des campagnes précédentes.

Le chargeur choisit une borne basse entre 5 Hz et mask_min, puis une borne haute entre mask_max et 80 Hz. Les lignes situées à l'extérieur de la bande conservée sont remplacées par −1. Un canal binaire supplémentaire peut indiquer quelles zones restent disponibles. Ce canal informe le réseau ; il ne retire pas automatiquement les pixels des opérations de convolution, et la cible reste le profil Vs complet.

La configuration actuelle prévoit deux variantes masquées et une version propre pour chaque fichier, soit trois exemples. Les bornes des masques sont déterminées à partir de l'indice et d'une graine dédiée, tandis que le bruit est tiré au chargement. Pour éviter qu'un même profil se retrouve dans plusieurs ensembles, la séparation doit se faire au niveau des fichiers avant de développer les variantes. La méthode indices_for_files est prévue pour cette opération.

L'ajout du canal binaire fait passer l'entrée de trois à quatre canaux et modifie donc la première convolution. Les anciens poids ne peuvent pas être rechargés directement dans cette version sans adaptation. Pour poursuivre les essais, il faudra comparer l'absence d'augmentation, le bruit seul et le masquage avec ou sans canal de validité, puis mesurer séparément l'apport des descripteurs, de Fourier et de FiLM.

Les difficultés rencontrées et la gestion du projet

La première difficulté a été de travailler dans un domaine qui n'était pas le mien. Je connaissais les principes du deep learning, mais je devais apprendre le vocabulaire des ondes et comprendre ce que les données représentaient. Les lectures et les échanges avec Dr. Kinh ont été nécessaires pour passer d'une vision purement informatique des tenseurs à une interprétation liée au dispositif de mesure.

La gestion des expériences a été un autre apprentissage. Le 24 juillet, j'ai constaté que les modèles utilisaient 90 000 données au lieu de 300 000. Cet épisode m'a montré qu'un nom de

dossier ne suffisait pas pour documenter un entraînement. Il faut conserver le nombre réel d'exemples, la configuration et, si possible, la liste des fichiers employés.

Tableau 6 : Difficultés de réalisation et réponses associées.

Difficulté	**Réponse mise en œuvre ou enseignement**
Coût des entraînements	Comparer sur un sous-ensemble et prioriser les hypothèses
Écart train et validation	Suivre les courbes, retenir une époch selon la validation et tester le bruit
Branche scalaire surdimensionnée	Compter les paramètres puis remplacer la représentation répétée
Panne électrique	Identifier le besoin de sauvegarde et de reprise ; main.py expose désormais resume=True
Données synthétiques et terrain	Limiter les conclusions de précision tant qu'une référence indépendante manque

Les temps de calcul ont également influencé mon organisation. Certains essais demandaient une longue attente, et je ne pouvais pas multiplier les modifications sans réfléchir à ce qu'elles permettraient de vérifier. Je préparais donc les expériences suivantes pendant les entraînements, tout en gardant du temps pour analyser les courbes. Une panne électrique a aussi interrompu un calcul, ce qui a renforcé l'intérêt des sauvegardes et de la reprise d'entraînement.

Enfin, la communication scientifique a représenté une part importante du travail. Préparer ma présentation m'a obligé à expliquer le problème avant de détailler les réseaux, puis à relier les résultats aux décisions prises. Après le stage, j'ai transmis à Kinh le dépôt GitHub, le code et les modèles, en mettant en avant DenseNet-121 et EfficientNet-B0 avec géométrie comme les variantes les plus intéressantes de mes essais. Cette transmission devait lui permettre de poursuivre les travaux à partir des résultats obtenus.

Conclusion

Ce stage de mobilité internationale m'a permis de découvrir le travail de recherche dans un laboratoire de génie civil, tout en mobilisant ma formation en intelligence artificielle. Pendant environ deux mois et demi, j'ai contribué à un projet dont l'objectif était de retrouver un profil de vitesse des ondes de cisaillement à partir de mesures réalisées en surface. Cette application m'a donné un cadre concret pour comprendre ce qu'un modèle de deep learning peut apporter, mais aussi ce qu'il faut vérifier avant d'interpréter ses résultats.

Bilan des résultats

Le travail a commencé par la prise en main du code et la comparaison des architectures. J'ai étudié les courbes d'apprentissage, testé l'ajout de bruit gaussien et travaillé sur la représentation de la géométrie d'acquisition. La refonte de cette branche constitue le résultat principal du stage : EfficientNet-B0 passe d'environ 37 millions à 6,21 millions de paramètres, avec une erreur de validation presque inchangée sur la campagne à 90 000 données.

DenseNet-121 avec géométrie obtient la plus faible erreur parmi les variantes principales de cette campagne. EfficientNet-B0 présente de son côté un modèle plus compact. Les essais de bruit n'ont pas établi de bénéfice général, mais ils ont aidé à préciser les questions à traiter ensuite : mieux représenter les perturbations, travailler sur les bandes incomplètes et évaluer les modèles dans d'autres conditions que celles de l'apprentissage.

Ces résultats répondent à une partie de la mission. Ils montrent qu'il est possible de réduire fortement le coût du modèle tout en conservant une précision proche sur les données synthétiques. La validation sur des mesures réelles, la répétition des essais et l'étude des incertitudes restent nécessaires pour aller plus loin. Les expériences à 300 000 données et le masquage fréquentiel constituent des prolongements de cette démarche.

Compétences acquises

Sur le plan technique, j'ai renforcé ma pratique de PyTorch, du transfert d'apprentissage et de l'adaptation de réseaux à une tâche de régression. J'ai appris à lire les architectures en termes de coût, à examiner les métriques et à relier une modification du code à une hypothèse expérimentale. Le diagnostic de la branche scalaire a été particulièrement formateur : le problème ne venait pas uniquement du choix du backbone, mais aussi de la manière dont une information simple était représentée.

J'ai également progressé dans l'organisation du travail. Avec un temps de stage limité et des entraînements coûteux, je devais choisir les essais utiles, conserver des traces et préparer la suite pendant les calculs. La présentation en anglais et les discussions avec mes encadrants

m'ont aidé à expliquer des choix techniques à des interlocuteurs qui n'avaient pas la même spécialité que moi.

Bilan personnel et interculturel

Cette mobilité a été ma première expérience dans un laboratoire, et j'en garde une impression positive. J'ai apprécié la liberté qui m'était laissée pour organiser mes horaires et mes journées à distance. Cette autonomie m'a permis de trouver ma manière de travailler, tout en me demandant de rester régulier dans le suivi des expériences.

Le rythme de l'équipe m'a aussi marqué. Certains membres travaillaient tard le soir, parfois toute la nuit, ce qui différait de mes habitudes en France. Cette observation concerne le laboratoire dans lequel j'ai été accueilli ; elle m'a surtout fait découvrir une autre organisation quotidienne du travail. Au-delà des modèles et des résultats, j'ai appris à m'adapter à un environnement nouveau, à poser des questions dans un domaine que je découvrais et à contribuer à un projet collectif.

Je retiens de ce stage une meilleure connaissance de la recherche appliquée et des possibilités qu'elle offre à un ingénieur en IA. Il a élargi ma vision des environnements dans lesquels mes compétences peuvent être utiles, en reliant directement l'informatique à une question physique et à un besoin de compréhension du sol.

Bibliographie

[1] Park, C. B., Miller, R. D. et Xia, J. (1999). Multichannel analysis of surface waves. Geophysics, 64(3), 800–808. DOI : 10.1190/1.1444590. Notice de l'USGS.

https://pubs.usgs.gov/publication/70021640

[2] National Yang Ming Chiao Tung University. About Us, NYCU ELITE. Présentation institutionnelle de la fusion des universités en 2021.

https://elite.nycu.edu.tw/about/

[3] National Yang Ming Chiao Tung University. Chih-Ping Lin, profil académique. Domaines de recherche et affiliation au département de génie civil.

https://scholar.nycu.edu.tw/en/persons/chih-ping-lin/

[4] Wu, T. J., Lin, C. P., Po, Y. M., Tran, Q. K., Pan, E. et Yang, S. H. (2025). Accurate non-destructive evaluation of pavement stiffness profile based on Monte Carlo inversion of surface-wave dispersion spectra. NDT & E International, 156, 103496.

https://doi.org/10.1016/j.ndteint.2025.103496

[5] Cho, S., Pyun, S., Choi, B., Lee, G., Jang, S. et Choi, Y. (2024). Prediction of S-wave velocity models from surface waves using deep learning. Near Surface Geophysics, 22, 281–297. Publication en ligne initiale le 21 novembre 2023.

https://doi.org/10.1002/nsg.12284

[6] Tran, Q. K., Lin, C. P., Wu, T. J., Cao, H. G., Pan, E. et Po, Y. M. (2025). Backcalculation of shear-wave velocity and thickness of asphalt pavement by deep-learning inversion of surface-wave dispersion spectra. International Journal of Pavement Engineering, 26(1), 2592682.

https://doi.org/10.1080/10298436.2025.2592682

[7] He, K., Zhang, X., Ren, S. et Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of CVPR, 770–778.

https://doi.org/10.1109/CVPR.2016.90

[8] Huang, G., Liu, Z., van der Maaten, L. et Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. Proceedings of CVPR, 4700–4708.

https://arxiv.org/abs/1608.06993

[9] Tan, M. et Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of ICML, PMLR 97, 6105–6114.

https://proceedings.mlr.press/v97/tan19a.html

[10] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. et Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. Proceedings of ICCV, 10012–10022.

https://arxiv.org/abs/2103.14030

[11] Perez, E., Strub, F., de Vries, H., Dumoulin, V. et Courville, A. (2018). FiLM: Visual Reasoning with a General Conditioning Layer. Proceedings of AAAI, 32(1).

https://doi.org/10.1609/aaai.v32i1.11671

[12] Zhu, W., Mousavi, S. M. et Beroza, G. C. (2019). Seismic Signal Denoising and Decomposition Using Deep Neural Networks. IEEE Transactions on Geoscience and Remote Sensing, 57(11), 9476–9488. Prépublication disponible sur arXiv.

https://arxiv.org/abs/1811.02695

Annexes

Annexe 1 : repères et organisation

Tableau 7 : Sigles et notations utilisés dans le rapport.

Terme	Signification
MASW	Analyse multicanal des ondes de surface
FVS	Spectre de dispersion utilisé en entrée du réseau
Vs / c	Vitesse de cisaillement / vitesse de phase
CNN / MLP	Réseau convolutif / perceptron multicouche
FiLM	Modulation affine des caractéristiques conditionnée par la géométrie
RMSE	Racine de l'erreur quadratique moyenne
x0 / dx / Ch	Déport de la source / espacement / nombre de capteurs
L	Longueur de la ligne de capteurs

Calendrier général du stage

Tableau 8 : Organisation générale des mission.

Période	Étape principale	Résultat attendu
Juin	Prise en main et lectures	Comprendre le problème et les données
Juillet	Instrumentation et comparaison	Disposer de courbes et de mesures comparables
Juillet et août	Essais de bruit	Évaluer une première augmentation
Fin août	Refonte géométrique	Réduire les paramètres et comparer les variantes
Fin de stage	Présentation et transmission	Permettre la poursuite du projet

Annexe 2 : code et calculs de paramètres

Le lien pour retrouver le code GitHub est le suivant :

https://github.com/SCL3/Backcalculation-of-shear-wave-velocity

Les explications techniques du rapport portent sur les cinq scripts de Model_100/300K_dataset_files : main.py, model.py, model_test_geo.py, Call_dataset.py et loss_fcns.py. Les autres dossiers permettent de situer les expériences dans l'organisation du projet. Les tableaux de résultats sont calculés à partir des journaux, sans relancer les entraînements.

Les fichiers de résultats exploités sont dans le dossier /log.

Calcul du coût de la géométrie

Tableau 9 : Budget de paramètres des branches géométriques, calculé à partir des couches.

Branche	Paramètres entraînables	Description
Initiale	30 845 568	14 516 → 2 048 → 512 → 128
Scalaire V2	4 290	BatchNorm puis 1 → 32 → 128
G1	9 216	13 descripteurs → 64 → 128
Encodeur avec Fourier	13 312	77 descripteurs → 64 → 128
EfficientNet géométrie complète	58 896	Encodeur avec Fourier + FiLM de rang 16

Pour la branche initiale, la première couche seule comporte 14 516 × 2 048 + 2 048 = 29 730 816 paramètres. Les deux couches suivantes ajoutent 1 049 088 et 65 664 paramètres, soit 30 845 568 au total. Le gain provient donc principalement de la suppression de cette première projection redondante.

Dans la branche finale d'EfficientNet, les couches de l'encodeur donnent 77 × 64 + 64 + 64 × 128 + 128 = 13 312 paramètres. Le générateur FiLM ajoute 128 × 16 + 16 + 16 × 2 560 + 2 560 = 45 584 paramètres. Le total de 58 896 doit être distingué du coût de la fusion et de la tête de prédiction.

La réduction totale est calculée par 1 − Pnouveau / Pancien. Pour EfficientNet, 1 − 6 211 313 / 36 997 985 ≈ 83,2 %. Ces nombres décrivent les modèles à trois canaux utilisés dans les journaux. Ajouter un canal à la première convolution change légèrement le total.

Annexe 3 : résultats complémentaires

Tableau 10 : Minima de val_loss selon l'écart type du bruit, campagne 90K.

Modèle	σ = 0	σ = 0,02	σ = 0,05	σ = 0,08	σ = 0,10
CNN ResNet-50 initial	0,063701	0,065256	0,068753	0,072156	0,077839
ResNet-50 révisé	0,062664	0,064746	0,066988	0,071912	0,079299
DenseNet-121	0,060956	0,062981	0,065692	0,070996	0,079087
Swin-T	0,066847	0,066571	0,068421	0,072312	0,074238
EfficientNet-B0	0,062671	0,064087	0,070461	0,076367	0,084391

Le tableau rassemble les minima de val_loss. Les cinq colonnes correspondent à des exécutions distinctes ; les nombres ne sont pas des moyennes sur plusieurs graines. ResNet-34 n'est pas inclus dans cette matrice de bruit, faute de série équivalente dans les journaux considérés.

Identification des expériences principales

Tableau 11 : Identifiants temporels des exécutions utilisées dans le bilan.

Expérience	Date et heure de lancement	Meilleure époch
CNN initial 90K	17/07/2026 15:59:40	165
DenseNet-121 initial 90K	22/07/2026 02:48:02	99
EfficientNet-B0 initial 90K	24/07/2026 03:04:44	169
DenseNet-121 + géométrie 90K	27/08/2026 06:09:42	142
EfficientNet-B0 + géométrie 90K	28/08/2026 00:05:16	184
DenseNet-121 + géométrie 300K	01/09/2026 03:56:16	134
EfficientNet-B0 + géométrie 300K	01/09/2026 20:59:51	147

Les nombres de paramètres sont repris de num_params. Les durées totales mentionnées dans le rapport sont obtenues en additionnant epoch_duration_sec, puis en divisant par 3 600. Elles excluent les intervalles non enregistrés et ne mesurent pas le temps consacré à la préparation, à la lecture des résultats ou à l'organisation des essais.

Dans les tableaux principaux, « meilleure époch » signifie minimum numérique de val_loss. Pour la campagne 300K, cette convention diffère de la dernière époch marquée is_best, comme indiqué dans le corps du rapport. Les lignes originales doivent être conservées pour retrouver le point de sauvegarde effectivement retenu.

Annexe 4 : configuration et reprise du projet

La version actuelle du point d'entrée contient des options plus récentes que les expériences initiales. Le tableau suivant distingue le protocole de référence de 90K et la configuration active dans le code consulté. Il ne prétend pas reconstruire à lui seul l'historique complet de chaque exécution.

Tableau 12 : Différence entre protocole historique et configuration actuelle.

Paramètre	**Campagne initiale 90K**	**Code actif consulté**
Exemples physiques	90 000 annoncés	MAX_FILES = 300 000
Canaux	3	4 avec canal de validité
batch	8	64
épochss maximales	200	250
Optimiseur	Adam	AdamW, weight_decay = 10^{-4}
Taux d'apprentissage	10^{-4}	Base 3 × 10^{-4} ; échauffement puis cosinus
Planification	Division par 2 toutes les 20 époch s	5 époch s d'échauffement ; horizon 120 ; plancher 10^{-6}
Arrêt anticipé	Patience 45	Patience 25 ; seuil relatif 10^{-3}
Augmentation	Sans bruit pour la référence	Bruit 0,02 + deux variantes masquées + une propre
Validation	20 %	20 %, augmentation désactivée
Reprise	Non déduite du protocole initial	resume=True dans l'appel

Pour reprendre une expérience, il faut conserver le commit du code, les paramètres, la liste des fichiers et la séparation entraînement-validation. Le jeu de test doit rester indépendant du choix des modèles. Si plusieurs géométries proviennent d'un même profil de sol, une séparation par profil source mérite aussi d'être examinée.

Les variantes masquées d'un même fichier doivent rester dans le même ensemble. Pour une validation propre, les augmentations restent du côté de l'entraînement ; un test sur des spectres incomplets doit définir séparément ses bornes et sa graine.

La reprise doit également vérifier les fichiers d'entrée, les normalisations et la compatibilité du nombre de canaux avec les poids. Des essais à plusieurs graines, une mesure du temps d'inférence et une évaluation physique des profils compléteraient le bilan : vitesses plausibles, interfaces, erreur selon la profondeur et cohérence du spectre recalculé.