Rapport de stage

Prédiction rapide des inondations de tsunamis par réseaux de neurones

Tom Sprunck

Encadrante : Audrey Gailler

2021

Remerciements

Je souhaite remercier Audrey Gailler pour la confiance qu'elle m'a accordé et pour son encadrement bienveillant durant ce stage.

Je remercie tous les stagiaires pour leur camaraderie au cours de ces six mois. Merci à Charly de s'être assuré de notre bonne intégration au sein du laboratoire, ainsi qu'à l'ensemble de l'équipe pour son accueil.

Table des matières

Introduction

La simulation des inondations causées par des tsunamis est un enjeu majeur dans les zones à risques. Des modèles non linéaires performants basés notamment sur les équations de Saint-Venant et de Boussinesq permettent de simuler la propagation des vagues au large et l'inondation des côtes. L'implémentation de ces modèles offre la possibilité de simuler la propagation des vagues d'un tsunami d'origine sismique à partir des paramètres de la source du séisme.

Cependant le calcul précis des hauteurs de vague à la côte nécessite une résolution de grille très fine pour prendre en compte la bathymétrie irrégulière. Malgré des progrès dans l'optimisation des codes (parallélisation, accélération sur GPU) au cours des trente dernières années, le temps de calcul reste trop prohibitif pour une utilisation opérationnelle. Les centres d'alerte nécessitent des outils de prévision rapide pour pouvoir estimer le niveau de danger et prévenir la population en cas de risque élevé. Le Centre national d'alerte aux tsunamis (CENALT), installé au sein du Commissariat à l'énergie atomique et aux énergies alternatives (CEA) de Bruyères-Le-Châtel a pour mission de surveiller l'apparition de tsunamis sur les côtes françaises en Méditerranée et Atlantique nord-est. Un message d'alerte doit être envoyé dans les quinze minutes suivant un séisme possiblement tsunamigène. Dans ce cadre, le temps de calcul d'une simulation en grille fine à la côte est trop important au vu du temps d'arrivée de la vague et se cumule avec le temps d'estimation des paramètres de source. L'utilisation de moyens de calculs massifs et la parallélisation des codes ne suffisent pas à dépasser cette limite.

Des outils de prévision rapide sont donc développés pour pallier ces contraintes temporelles et fournir des solutions aux centres d'alerte [5, 2]. L'utilisation d'algorithmes d'apprentissage, et en particulier de réseaux de neurones commence à se développer dans ce domaine depuis quelques années. Les réseaux de neurones proposent un temps d'évaluation court (souvent inférieur à la seconde) qui s'avère intéressant dans le contexte d'estimation rapide.

Deux approches de prévision par réseaux de neurones ont été utilisées par le passé. La première et la plus commune se base sur une architecture de type *multilayer perceptron* (MLP) et vise à transformer une grille grossière de hauteurs de vagues maximales au large en une prédiction des hauteurs maximales en grille fine au niveau de la côte. Plusieurs études comparatives sur les côtes japonaises [4, 21] ont montré l'efficacité de cette méthode pour prédire des inondations importantes en obtenant des résultats similaires aux modèles non linéaires. La deuxième méthode étudiée [4] repose sur un réseau de neurones convolutif (CNN). Le problème habituel de régression pour l'estimation de la hauteur de vague est remplacé par un problème de classification qui vise à trouver le scénario le plus proche de la situation considérée dans une base de donnée précalculée, toujours en se basant sur une simulation basse résolution.

Les réseaux de neurones semblent encore sous-exploités pour la prévision des inondations de tsunamis. Le but de ce stage est de développer les approches déjà testées. Les travaux s'appuient sur un stage précédent visant à implémenter un MLP pour la prévision d'inondations sur la ville de Cannes [1]. Une seconde base de données développée dans le cadre d'un projet similaire et basée au niveau de Nice permettra d'évaluer la généralité des algorithmes étudiés. On note que ces bases de données diffèrent des bases utilisées dans les études menées au Japon. Les contextes sismotectoniques des zones étudiées sont très différents. Le bassin méditerranéen est caractérisé une sismicité modérée (magnitudes inférieures à 7.5), tandis que le Japon présente une sismicité forte avec des magnitudes allant jusqu'à 9. Les amplitudes de vagues de tsunami sont par conséquent très différentes et peuvent varier de $20cm$ à $4m$ en méditerranée occidentale contre 3 à $40m$ au niveau des côtes japonaises. L'impact à la côte et l'étendue des inondations sont donc difficilement comparables. La résolution de grille fine souhaitée est ici beaucoup plus élevée (de l'ordre de $10m$), ce qui nécessitera d'utiliser des grilles de sortie et d'entrée de dimensions plus importantes si l'on souhaite conserver cette précision. Il s'agira d'étudier l'influence des différents paramètres sur l'efficacité des modèles et d'évaluer ces derniers.

1 Quelques rappels sur les réseaux de neurones

1.1 Perceptron multi-couches

Le premier modèle considéré est celui du perceptron multi-couches ou *Multi Layer Perceptron* (MLP). Un tel modèle est composé par plusieurs couches contenant chacune plusieurs neurones (*units*). On parle aussi de réseau dense ou *fully-connected*, car chaque neurone d'une couche donnée est connecté à tous les neurones de la couche précédente et de la couche suivante, imitant le fonctionnement des connexions cérébrales. Les couches situées entre les couches d'entrée et de sortie sont généralement appelées "couches cachées". Au cours de la phase d'apprentissage, le réseau va apprendre la correspondance entre un échantillon d'entrées X (*inputs*) et un échantillon de sorties correspondant Y (*ground truth*) qui lui sont fournis. Au terme de l'apprentissage, le modèle est capable d'établir une prédiction $\hat{y}$ (grille fille) à partir d'une entrée x (grille mère) qui ne lui a pas encore été présentée.

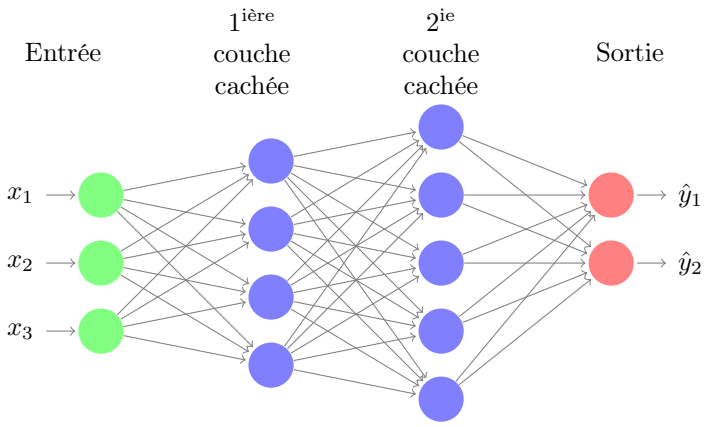


FIGURE 1 – Réseau de neurones MLP à deux couches cachées

Concrètement, un réseau MLP se présente comme une succession de modèles linéaires, dont on brise la linéarité par l'ajout de fonctions d'activation non linéaires. Les coefficients des modèles linéaires sont autant de poids affectés aux neurones du réseau et sont ajustés au cours de la phase d'apprentissage. Si l'on considère une couche à m neurones, prenant en entrées des vecteurs x de dimension n, la sortie d'un neurone de cette couche peut s'écrire :

$$\hat{y}_i = \sigma\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right), \quad 1 \leq i \leq m \tag{1}$$

où w est la matrice du modèle linéaire, b un vecteur de biais et σ la fonction d'activation utilisée. Les coefficients w_{ij}, $1 \leq j \leq n$ correspondent aux poids affectés aux connexions du neurone i avec les n entrées (par exemple les sorties des neurones d'une couche précédente). Sous forme matricielle, un réseau MLP comportant une couche cachée à m_1 neurones et une couche de sortie à m_2 neurones (c'est-à-dire que le vecteur de sortie est de dimension m_2) peut s'écrire :

$$\hat{y} = \tilde{\sigma}\left(\tilde{w}.\sigma(w.x + b) + \tilde{b}\right) \tag{2}$$

où w et $\tilde{w}$ sont les matrices des poids correspondants à la couche cachée et la couche de sortie, de tailles respectives $n \times m_1$ et $m_2 \times m_1$.

Les fonctions d'activation permettent de mettre en valeur ou d'atténuer les connexions en sortie d'une couche donnée. L'une des fonctions les plus fréquemment utilisées est la fonction "Relu", pour *Rectified Linear Unit* :

$$\sigma : x \mapsto \begin{cases} x & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases} \tag{3}$$

Les poids et les biais sont généralement initialisés aléatoirement, puis sont ajustés au cours de l'apprentissage par un algorithme d'optimisation. Une fonction coût renseignant sur l'erreur entre les prédictions et les valeurs réelles est minimisée au cours de l'apprentissage en modifiant les poids du modèle. L'une des fonctions coûts les plus courantes et qui sera utilisée ici est l'erreur quadratique moyenne ou MSE (*mean squarred error*) :

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4}$$

À chaque étape d'entraînement l'erreur est calculée par la propagation du signal d'entrée vers la sortie (passage *forward*) selon les formules précédentes, puis le gradient de l'erreur selon chacun des coefficients du modèle est calculé par rétropropagation. Ces gradients sont enfin utilisé par l'algorithme d'optimisation pour mettre à jour les poids. L'entraînement se poursuit ainsi tant qu'un critère d'arrêt (par exemple un nombre d'itérations maximal) n'a pas été atteint.

L'algorithme d'optimisation utilisé ici est l'algorithme Adam [15]. Il s'agit d'une méthode de descente de gradient se basant sur une estimation des moments d'ordre 1 et 2 du gradient. L'étape k d'optimisation d'une fonction f sur la variable x est donnée par les formules suivantes :

$$\begin{array}{ll} a) & g_k \leftarrow \nabla f(x_{k-1}) \\ b) & m_k \leftarrow \beta_1 m_{k-1} + (1-\beta_1) g_k \\ c) & v_k \leftarrow \beta_2 v_{k-1} + (1-\beta_2) g_k^2 \\ d) & \hat{m}_k \leftarrow \frac{m_k}{1-\beta_1^k} \\ e) & \hat{v}_k \leftarrow \frac{v_k}{1-\beta_2^k} \\ f) & x_k \leftarrow x_{k-1} - \alpha \frac{\hat{m}_k}{\sqrt{\hat{v}_k + \epsilon}} \end{array} \tag{5}$$

β_1 et β_2 sont des paramètres réels compris dans l'intervalle $[0, 1[$ et sont en général initialisés à 0.9 et 0.999. α correspond au taux d'apprentissage, souvent fixé à 0.001. Les étapes b) et c) procèdent au calcul des moyenne et variance du gradient selon les itérations passées. Les moments m_k et v_k sont initialisés à zéro. Les étapes d) et e) servent à contrer le biais de cette initialisation dans les premières itérations, β_1^k et β_2^k convergeant rapidement vers 0 au cours des itérations. La mise à jour de la variable x est effectuée en étape f), où ϵ désigne un terme de lissage utile en cas de très faible variance $\hat{v}_k$. Concrètement dans notre cas d'application la variable x est un vecteur de dimension importante contenant tous les poids du réseau, et f es la fonction coût considérée.

Un avantage des réseaux MLP est qu'il sont très versatiles et nécessitent peu de pré-traitement. En notant $M_x \times M_y$ et $N_x \times N_y$ les dimensions des grilles d'entrée et de sortie, les grilles mère et fille sont simplement aplaties en des vecteurs de longueur $M_x.M_y$ et $N_x.N_y$. Le réseau transforme chaque vecteur de longueur $M_x.M_y$ en un vecteur de taille $N_x.N_y$ qui est ensuite redimensionné en une grille $N_x \times N_y$ pour la visualisation des résultats et le post-traitement.

Si le postulat de base est très simple, de nombreux facteurs peuvent influencer les performances du modèle et seront étudiés en section 3. Le choix des grilles et des hyperparamètres du modèle figurent parmi les facteurs les plus influents. En particulier, il n'est pas nécessaire de prédire la hauteur des vagues à chaque point de la grille de sortie : les points correspondants à des profondeurs ou élévations élevées peuvent être ignorés. Une amélioration simple du modèle est également possible et à tester : l'ajout des paramètres de source en entrée du modèle.

1.2 Réseaux de convolution

L'une des faiblesses des réseaux de type MLP est l'absence intrinsèque de localité dans le traitement des images d'entrée. En effet, les grilles sont aplaties dès l'entrée dans le réseau, et avant l'entraînement ce dernier ne fait aucune différence entre la connexion entre deux pixels proches ou éloignés. C'est l'inconvénient des réseaux denses où tous les neurones de deux couches consécutives sont connectés.

Les réseaux de convolution présentent une approche différente. L'idée est, pour un pixel donné, d'effectuer une moyenne pondérée des valeurs contenues par ce pixel et des pixels voisins pour calculer la valeur du pixel correspondant sur une nouvelle image. De plus, cette opération peut être réalisée en parallèle sur plusieurs

images superposées, et permet d'obtenir une image à une ou plusieurs couches. Par exemple, une image couleur se représente en fait par une superposition de trois matrices qui correspondent aux niveaux de rouge, vert et bleu (image RGB). Dans notre cas d'application, on pourrait par exemple superposer les hauteurs de vague minimales et maximales pour obtenir une image à deux "couleurs" (on parle aussi de filtres ou de canaux). Si l'on considère des grilles de dimension $N_y \times N_x$, le tenseur correspondant serait de dimension $N_y \times N_x \times 2$. Cette possibilité sera explorée en section 6.1.

Si l'on considère une couche de convolution prenant en entrée une image à n filtres et produisant une image à m canaux, le calcul peut s'écrire :

$$\hat{y}_{i,j,k} = \sigma\left(\sum_{d_i,d_j,l} w_{d_i,d_j,l,k}.x_{i+d_i,j+d_j,l} + b_k\right) \tag{6}$$

où w est le noyau de convolution, un tenseur de dimension $H \times W \times n \times m$ que l'on peut voir comme autant de petites matrices $H \times W$ que l'on applique le long de chaque filtre de l'image d'entrée. On choisit le plus souvent $H = W$, en prenant des valeurs impaires assez petites (généralement entre 3 et 11). L'opération est alors définie sans ambiguïté, le noyau pouvant être centré en chaque pixel, ce qui n'est pas le cas lorsque la matrice possède une dimension paire. Le traitement des bords de l'image (pour lesquels le noyau va sortir de l'image) est fait en ajoutant des zéros où cela est nécessaire (*zero-padding*), ce qui permet de conserver la taille de l'image, ou bien en effectuant le calcul uniquement pour les pixels situés suffisamment à l'intérieur de l'image. La taille de l'image est alors réduite de $2\frac{H-1}{2} = H - 1$ pixels en hauteur et $W - 1$ pixels en largeur. La figure 2 présente un exemple de convolution : les "pixels" situés dans la zone orange de la matrice de départ sont connectés au "pixel" orange du résultat par l'application du noyau de convolution (en bleu).

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 6 & 7 & 6 \\ 1 & 5 & 9 & 6 \\ 5 & 4 & 4 & 9 \\ 3 & 5 & 7 & 4 \end{pmatrix}$$

FIGURE 2 – Exemple de convolution entre deux matrices

On peut noter qu'ici un pixel de l'image de sortie n'est connecté qu'à un nombre réduit de pixels de l'image de départ. Si l'on considère une couche de convolution, chaque pixel de sortie est connecté à $H.W$ pixels de chaque filtre de l'image d'entrée. Si l'on effectue plusieurs convolutions successives, cette zone de connectivité s'étend.

De la même façon que les réseaux MLP concentrent l'information dans les couches cachées, les réseaux de convolution vont encoder les composantes de l'image d'entrée dans les différents filtres. On accompagne généralement les couches de convolutions par des opérations de compression qui réduisent les dimensions de l'image. L'opération la plus couramment utilisée est le *max-pooling*, qui consiste simplement à garder localement le pixel d'intensité maximale. On choisit souvent pour cela un noyau de taille 2×2 : on garde localement un pixel parmi quatre, ce qui revient à diviser la dimension de l'image par deux dans chaque direction. Une autre option est de réduire la taille de l'image directement durant la convolution, en augmentant l'espacement (*stride*) entre deux applications du noyau. On peut par exemple appliquer le noyau tous les deux pixels, ce qui aura pour conséquence de diviser la dimension par deux. Le résultat obtenu sera plus lisse qu'une opération de *max-pooling*.

Enfin, une conséquence importante de la réduction des dimensions de l'image est l'élargissement de la zone de connectivité de chaque pixel de l'image de sortie. L'évolution de l'étendue de la zone de connectivité dans une direction après une convolution est donnée par la formule suivante :

$$c_{i-1} = k + s.(c_i - 1) \tag{7}$$

où c_i désigne le nombre de pixels de la i^{e} image connectés à un pixel donné de l'image finale, k la dimension du noyau et s l'espacement. On peut ainsi calculer facilement le nombre de pixels de l'image initiale connectés

à un pixel de l'image finale après l'application de plusieurs convolutions. Les figures 3 et 4 représentent ce processus en une dimension pour simplifier la visualisation. Les cellules colorées sont connectées à la cellule rouge de la dernière image obtenue, notée I_n. Les cellules hachurée correspondent aux centres d'applications du noyeau. Les flèches relient une cellule d'application du noyeau à la cellule générée par cette application sur l'image suivante. On retrouve les étendues données par la formule 7.

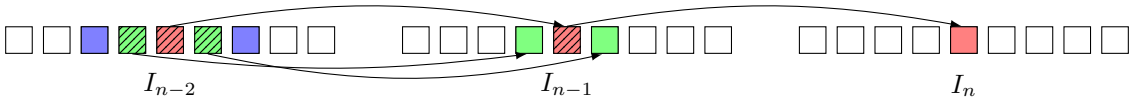


FIGURE 3 – Extension de la connectivité pour des convolutions successives avec un noyau de taille 3 (*stride*=1)

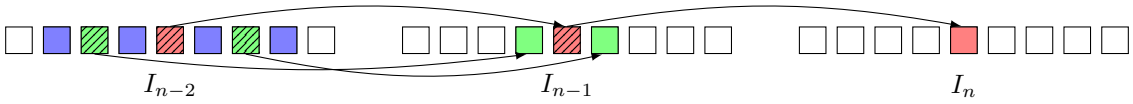


FIGURE 4 – Extension de la connectivité pour des convolutions successives avec un noyau de taille 3 (*stride*=2)

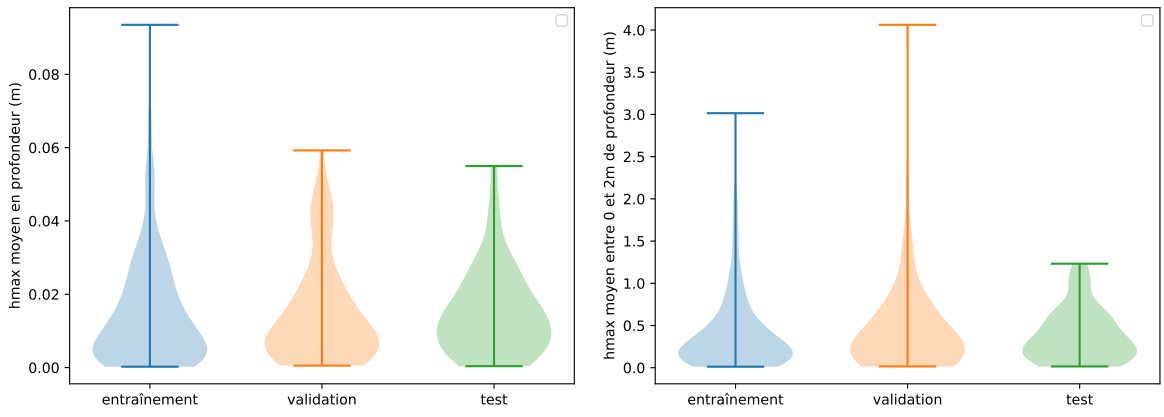
2 Découpage du jeu de données et choix des grilles

Deux sites d'étude ont été principalement considérés, localisés sur les villes de Cannes et Nice. La base de donnée utilisée pour Cannes contient près de 400 simulations, tandis que celle de Nice en contient plus de 1500. Un dernier jeu de données localisé sur Antibes a été généré en fin de stage et est basé sur les mêmes scénarios que celui de Cannes.

2.1 Décomposition des données

Les scénarios constituant les bases de données présentent des sources sismiques issues de quatre zones géographiques différentes : la marge nord-algérienne (zones $3, 4$), la mer Ligure (zone 5) et l'ouest de l'Italie (zone 6). Ces scénarios sont issus d'une évaluation probabiliste du risque tsunami en méditerranée occidentale [25] se basant sur la base de failles du CENALT. On peut ainsi découper chaque base de données en plusieurs catégories. Pour le site de Cannes on retient quatre catégories : deux pour les séismes originaires respectivement des zones 5 et 6 et deux autres contenant les séismes des zones 3 et 4 mélangés, une catégorie contenant des scénarios de faible amplitude et l'autre des scénarios de plus forte amplitude. Dans le cas de Nice, les catégories choisies correspondent simplement aux quatre zones, le jeu contenant davantage de simulations. Les simulations sont générées à l'aide du code de calcul Taitoko développé au CEA [8], en résolvant les équations de Saint-Venant non linéaires sur un jeu de quatre grilles imbriquées à résolutions croissantes.

Le but recherché est d'assurer une bonne variété de scénarios dans les données d'entraînement et de validation : particulièrement dans le cas de la base de données de Cannes, un découpage aléatoire malheureux pourrait amener à une sur ou sous représentation de certains types de scénarios dans les jeux d'entraînement et de validation. On extrait 10% de chacune des catégories pour remplir un jeu de données test. Lors de l'entraînement, on procède à une forme d'échantillonnage stratifié : pour un ratio de validation de 20% on extrait environ 20% de chaque catégorie pour remplir le jeu de validation. Les données sont finalement mélangées avant l'entraînement. La répartition des données dans chaque jeu est représentée pour Cannes en figure 5. La figure 5a compare les distributions des hauteurs maximales moyennées sur l'ensemble des cellules immergées des grilles mères. La figure 5b effectue la même comparaison pour les grilles filles en se concentrant sur les cellules situées proches de la côte (entre 0 et $2m$ de profondeur). On observe des distributions assez semblables pour les trois jeux de données. Le jeu de validation semble en particulier contenir suffisamment de scénarios à forte amplitude, ce qui sera important pour sélectionner un modèle performant au niveau des inondations.



(a) Grilles mères

(b) Grilles filles

FIGURE 5 – Distributions des hauteurs maximales pour le site de Cannes

2.2 Traitement des grilles et assemblage du jeu de données

Les résultats des simulations sont stockés dans des fichiers .grd, chaque fichier correspondant à une grille mère ou fille. Le langage de programmation utilisé pour les traitements et entraînements est *Python*. Pour

simplifier les traitement suivants, les fichiers sont lus à l'aide de la bibliothèque *rasterio* puis les matrices sous-jacentes sont extraites et converties en archives *numpy* .npz, ce qui permet d'obtenir un unique fichier par simulation. Les transformations affines permettant la conversion de coordonnées en indices matriciels sur les grilles mère et fille sont également stockées sous ce format, et une classe personnelle implémentant ces conversions sera utilisée par la suite, sans nécessiter d'autres librairies que *numpy*.

Chaque catégorie de scénario est contenue dans un dossier séparé. Les jeux de test et de validation sont également isolés dans d'autres dossiers. Pour pouvoir stocker directement les données d'entrée dans la RAM et limiter les traitements au moment de l'apprentissage, les grilles d'entraînement et de validation sont préalablement découpées et regroupées en quatre fichiers *numpy* .npy correspondant aux matrices contenant les sous-grilles d'entrée et de sortie. Ces sous-grilles sont détaillées dans la section suivante.

Pour limiter la quantité de paramètres des réseaux utilisés et faciliter leur apprentissage, on peut également choisir d'ignorer certains points des grilles d'entrée et de sortie selon leur profondeur. Un choix naturel est par exemple de supprimer les points de profondeur nulle des grilles d'entrées, les inondations n'étant pas prédites sur celles-ci. On peut également réduire les grilles filles en limitant les prédictions aux points situés à faible profondeur pour ne regarder que les points proches de la côte. On essaiera en particulier de prédire sur Cannes les hauteurs entre une profondeur de $-50m$ et une altitude de $8m$.

2.3 Choix des grilles pour Cannes et Antibes

Pour le site de Cannes, les grilles mères originelles (de résolution $3700m$) fournies par les simulations sont de dimensions 331×691, soit 228721 noeuds, et couvrent le bassin méditerranéen occidental. Les grilles filles, de résolution $10m$, sont elles de taille 1301×1731, soit 2252031 noeuds. À titre de comparaison, une étude portant sur les très fortes inondations dans la région de Nankai [4] utilise des grilles basse et haute résolutions contenant respectivement 400 et 5826 cellules. La résolution de la grille de sortie est dans ce cas de l'ordre de $30m$. Il est nécessaire d'extraire des sous-grilles des grilles mères et filles pour limiter la dimension des entrées et sorties des modèles. On note qu'il sera impossible d'obtenir un aussi petit nombre de variables de sortie dans notre cas d'application si l'on souhaite conserver une précision de $10m$ et une étendue spatiale du même ordre.

Le choix des grilles d'entrée et de sortie, et en particulier de leurs dimensions et de leurs localisations par rapport au domaine d'intérêt risque d'avoir une importance cruciale dans les performances du modèle. Il est nécessaire de choisir une grille d'entrée suffisamment étendue pour que le modèle possède suffisamment d'information pour inférer la hauteur de vague sur la grille fine. Le changement de résolution est brutal : une maille de la grille grossière a une taille typique allant de 900m (pour Nice) à quelques kilomètres (pour Cannes), contre une dizaine de mètres en grille fine. Une cellule de la grille mère correspond donc au moins à 90^2 cellules de la grille fille, la zone prédite par un modèle ne correspondra ainsi spatialement qu'à une poignée de cellules de la grille mère. Le modèle va donc devoir apprendre les correspondances entre le comportement global des vagues au large et la hauteur de vague à proximité de la côte, ce qui devra être pris en compte dans les architectures de réseaux considérées.

On doit également restreindre la taille des grilles de sortie pour ne contenir que l'information nécessaire à l'apprentissage et au post-traitement, sans surcharger le modèle. Ceci peut être effectué en redécoupant la grille fille en sous-grilles, puis en sélectionnant les cellules restantes selon leur localisation ou leur profondeur. On relève toutefois que la zone prédite par cette approche reste spatialement limitée si l'on souhaite conserver une précision de $10m$, le nombre de paramètres augmentant rapidement avec le nombre de variables de sortie. En effet le nombre de paramètres associés à une couche est le produit des dimensions d'entrée et de sortie de cette couche, auquel on additionne la dimension de sortie (ajout du terme de biais). Ainsi si l'on considère une grille de sortie de dimensions 128×256 (étendue spatiale de l'ordre de $3km^2$) et un réseau dont la dernière couche cachée contient 512 neurones, la couche de sortie comporte à elle seule près de 17 millions de paramètres entraînables. Pour la ville de Cannes, on reprend la grille de sortie utilisée dans le stage précédent [1] et l'on y ajoute deux grilles situées à l'est et à l'ouest de celle-ci (voir la table 7 en annexe et la figure 6).

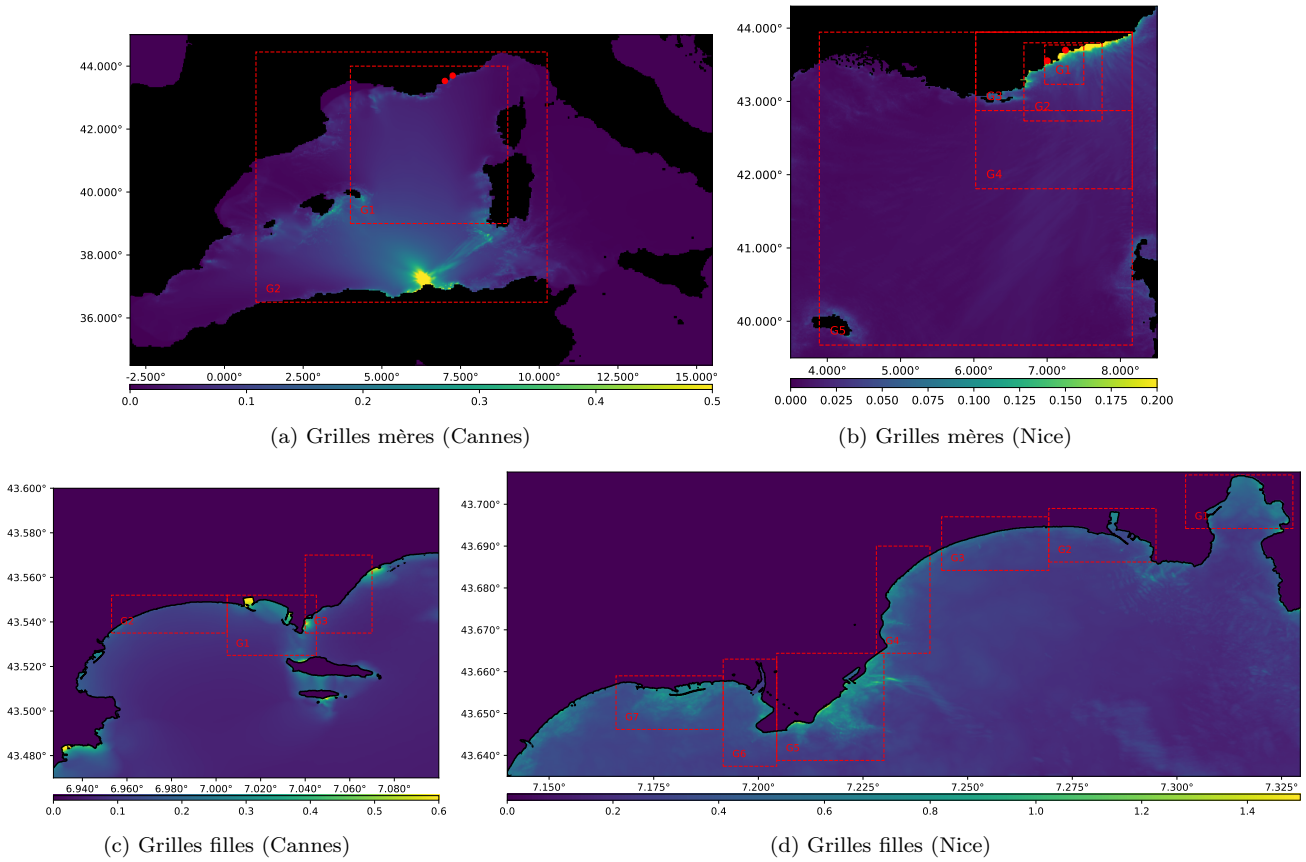
Deux grilles d'entrée ont été testées sur ce site. La première est similaires à celle originellement choisie dans le cadre du stage précédent, de dimensions 150×150. La seconde grille se veut plus large et s'étend jusqu'à la côte africaine. On réduit également le nombre de variables d'entrée en ignorant les points non immergés de la grille, le modèle utilisé pour les simulations ne calculant pas les inondations sur les grilles mères. Les dimensions des grilles d'entrée et de sortie pour la ville de Cannes sont résumées en tableau 7.

La figure 6 présente l'emprise des grilles d'entrée et de sortie retenues pour Nice et Cannes. La figure 51 en annexe présente le découpage effectué sur Antibes. On a tracé les points situés au dessus du niveau de la mer en noir dans les grilles mères (il n'y a pas de calculs d'inondations dans les grilles mères). On note une quantité importante de variables d'entrée et de sortie du modèle, même dans le cas des grilles tronquées par profondeur.

2.4 Choix des grilles pour Nice

Le second jeu construit pour Nice a été généré en utilisant des grilles d'entrée à une résolution de $900m$, ce qui permet d'augmenter la précision des simulations sur la grille fine au prix d'un léger surcoût de calcul pour la grille grossière. Les tailles des matrices des grilles mères et filles complètes sont alors respectivement 1273×2377 et 728×1896. Dans un second temps, les grilles d'entrée pour Cannes seront regénérées en prenant également un pas de $900m$, ce qui permettra de mesurer l'impact de la résolution sur la qualité des prédiction (section 5.3).

Le choix de sous-grilles de sortie pour la ville de Nice s'est voulu plus exhaustif en se basant sur l'expérience acquise sur Cannes. On a donc choisi sept sous-grilles couvrant une grande partie de la côte. Les sous-grilles sont de dimensions 128×256, 256×128 ou 256×256 pour s'adapter à la topologie locale. Chacune des dimensions est un multiple de deux, pour faciliter un traitement futur par des réseaux de convolution ou des redimensionnements. On extrait de la même façon plusieurs grilles d'entrée centrées sur la région, en allant d'une dimension de 64×64 à 256×256. On note que ces nouvelles grilles sont beaucoup plus resserrées géographiquement que les grilles utilisées pour Cannes, la résolution de la grille mère sous-jacente étant supérieure. La grille la plus large couvre une surface de 512×512 cellules soit environ $461 \times 461 = 212521$ kilomètres carrés. Elle est finalement redimensionnée au format 256×256 pour limiter la taille des entrées, au prix d'une perte de qualité. Les autres grilles sont directement tronquées des grilles obtenues par les simulations.



(a) Grilles mères (Cannes)

(b) Grilles mères (Nice)

(c) Grilles filles (Cannes)

(d) Grilles filles (Nice)

FIGURE 6 – Sous-grilles d'entrée et de sortie pour les sites de Cannes et Nice

3 Architecture MLP

Le but de cette section est de mettre en place un réseau de neurones de type *multilayer perceptron* pour la prédiction des hauteurs maximales de vague, et d'ajuster ses paramètres. Toute l'étude de paramètres se base sur le premier jeu de données disponible, basé sur Cannes.

L'architecture du modèle MLP implémenté est particulièrement simple. Nous considérons une couche d'entrée, suivie par plusieurs couches cachées possédant un même nombre fixé de neurones, puis enfin une couche de sortie. Les couches intermédiaires sont toutes suivies par une activation de type RELU. Des opérations de régularisation ont également été ajoutées pour limiter le sur-apprentissage (*overfitting*) et sont détaillées dans la section 3.3. Différents prétraitements sont testés en parties 3.2 et 3.4. Une recherche de meilleurs hyperparamètres est effectuée et permet la sélection d'un modèle, qui est évalué en partie 3.7.

3.1 Implémentation

L'implémentation des réseaux de neurones est effectuée à l'aide des librairies *Tensorflow* et *Keras*. Dans un premier temps, un script d'entraînement généraliste a été développé. Le but était de pouvoir le paramétrer au maximum directement depuis la ligne de commande à l'aide de nombreux arguments optionnels (composition des couches du réseau, traitements ...). Le chargement des données était effectué à l'aide de fichiers de configurations personnalisés contenant les chemins vers les données, ainsi que les informations nécessaires à la visualisation (coordonnées de segmentation des grilles) et les options de découpage selon la profondeur si nécessaire.

Cependant, en multipliant les études systématiques de paramètres, l'utilisation d'un unique script multi-usages s'est révélé lourd à utiliser et à entretenir. Ce script généraliste a donc été remplacé par une multitude de scripts d'entraînement, de calculs d'erreur et de visualisation spécialisés, au risque d'effectuer des répétitions dans le code.

3.2 Réduction de dimension

La dimension des grilles d'entrée (même redécoupées) étant importante, il peut être intéressant de prétraiter les données d'entrée afin de réduire le nombre de variables fournies au modèle. Même en gardant uniquement les points de profondeur non nulle le volume de variables d'entrée reste très important (19198 pour la plus petite grille mère pour Cannes), beaucoup plus important que la taille de la base de données elle-même dans le cas du site de Cannes. Tronquer les grilles mère et fille selon la profondeur ne semble pas avoir d'impact significatif sur les performances des réseaux, en dehors d'une l'accélération de l'entraînement. Les premières études de paramètres ayant été effectuées avec une capacité de calcul réduite, on tronquera tout de même dans cette partie les grilles d'entrées aux points de profondeur négative, et les grilles de sortie aux points compris entre $-50m$ et $+8m$.

Une grande partie des variables d'entrées n'ajoute pas d'information utile à l'apprentissage du modèle. On peut donc appliquer une méthode de réduction de dimension pour réduire la complexité du modèle et faciliter son apprentissage. Cette réduction de dimension rajoute un niveau de difficulté au paramétrage du modèle, cette réduction pouvant nécessiter un paramétrage elle-même. L'efficacité d'une réduction de dimension est maximale lorsque le nombre de variables retenues est proche de la dimension intrinsèque du jeu de données. Déterminer la dimension intrinsèque d'un jeu de données est cependant un problème complexe [28] qui ne sera pas abordé. Quelques valeurs seront testées et comparées. Deux méthodes seront étudiées ici : la décomposition en composantes principales et l'utilisation d'un réseau auto-encodeur.

3.2.1 Analyse en composantes principales (ACP)

L'un des objectifs de la technique de décomposition en composantes principales [14] est de réduire la dimension d'un jeu de données comprenant de nombreuses variables corrélées tout en conservant autant que possible la variabilité des données d'origine. Les variables originelles sont transformées en un nouveau jeu de variables, appelées composantes principales. Ces nouvelles variables sont décorrélées et ordonnées de sorte à ce que les premières composantes principales traduisent un maximum de variabilité de l'ensemble des variables d'origine.

Le principe est le suivant : on considère un vecteur aléatoire X de dimension m. On souhaite en déduire un vecteur aléatoire de dimension n petite devant m qui conserve l'information contenue par les variances et covariances de X. On commence par chercher une combinaison linéaire des composantes de X possédant une variance maximale :

$$z_1 = \sum_{i=1}^{m} a_{1i} x_i = a_1^T.X \tag{8}$$

z_1 constitue la première composante principale. On cherche ensuite une combinaison linéaire $z_2 = a_2^T.X$ décorrélée de z_1 possédant une variance maximale. On peut ainsi construire de manière itérative jusqu'à m composantes principales.

Le calcul des composantes principales se base sur la matrice de covariance Σ de X. On peut montrer que la composante principale d'ordre k est donnée par $z_k = a_k^T.X$, avec a_k le vecteur propre correspondant à la k^{ie} plus grande valeur propre de Σ (on rappelle que Σ est symétrique, semi-définie positive).

En pratique les composantes principales s'obtiennent en utilisant la matrice de covariance empirique $\hat{\Sigma}$ du jeu de données. On note $\tilde{X} \in \mathcal{M}_{N,m}(\mathbb{R})$ une matrice contenant N réalisations du vecteur aléatoire, centrée selon sa moyenne empirique : $\tilde{x}_{ij} = x_{ij} - \bar{x}_j$. Alors $\hat{\Sigma}$ est définie par $\hat{\Sigma} = \frac{1}{N-1}\tilde{X}^T\tilde{X}$, et l'on obtient les composantes principales empiriques par le calcul des valeurs propres de $\hat{\Sigma}$. On note a_k les vecteurs propres de la matrice de covariance empirique et A la matrice $N \times p$ dont les colonnes sont les a_k. On a alors la relation

$$Z = \tilde{X}A \tag{9}$$

où Z désigne la matrice $N \times p$ des scores des composantes principales empiriques. Le score pour l'observation i sur la composante principale k est ainsi donné par : $z_{ik} = \sum_j a_{jk}.x_{ij}$.

L'implémentation utilisée (librairie *Scikitlearn*) utilise une décomposition en valeurs singulières. Toute matrice rectangulaire $\tilde{X} \in \mathcal{M}_{N,m}(\mathbb{R})$ admet une décomposition :

$$\tilde{X} = UDA^T \tag{10}$$

où, en notant r le rang de X :

- $U \in \mathcal{M}_{N,r}(\mathbb{R})$, $A \in \mathcal{M}_{r,m}(\mathbb{R})$ sont des matrices dont les colonnes sont orthonormées ($U^TU = \mathbb{I}_r = AA^T$)
- $D \in \mathcal{M}_{r,r}(\mathbb{R})$ est une matrice diagonale dont la diagonale contient les valeurs propres de $\tilde{X}^T.\tilde{X}$.

La preuve de cette égalité découle de la décomposition spectrale de $\tilde{X}^TX$:

$$(N-1)\hat{\Sigma} = \tilde{X}^T\tilde{X} = \sum_{i=1}^{r} \mu_i a_i.a_i^T \tag{11}$$

où les μ_i, a_i désignent les valeurs et vecteurs propres orthonormés de $\tilde{X}^T\tilde{X}$, les valeurs propres étant ordonnées par ordre décroissant. On définit A comme étant la matrice dont les colonnes sont les a_k, et U la matrice dont les colonnes sont les $u_k = \frac{1}{\sqrt{\mu_k}}\tilde{X}a_k$, $1 \leq k \leq r$. D est la matrice diagonale de coefficients les $\sqrt{\mu_k}$. On vérifie que $\tilde{X} = UDA^T$. Par définition :

$$DA^T = \begin{pmatrix} \sqrt{\mu_1}a_1^T \\ \vdots \\ \sqrt{\mu_r}a_r^T \end{pmatrix} \tag{12}$$

et

$$\begin{aligned} UDA^T &= \sum_{k=1}^{r} \frac{1}{\sqrt{\mu_k}}\tilde{X}a_k\sqrt{\mu_k}a_k^T \\ &= \sum_{k=1}^{r} \tilde{X}a_k a_k^T \\ &= \sum_{k=1}^{m} \tilde{X}a_k a_k^T \end{aligned} \tag{13}$$

On trouve finalement

$$UDA^T = \tilde{X} \sum_{k=1}^{m} a_k a_k^T = \tilde{X} \tag{14}$$

par orthonormalité. On note que la matrice U fournit une version mise à l'échelle des scores $z_{i,k}$ de sorte à obtenir une variance inférieure à $N-1$.

L'application d'une ACP aux entrées d'un réseau de neurones a déjà été explorée dans la littérature. On peut citer par exemple une application dans la classification d'objets sous-marins par analyse de données sonar [3], pour laquelle l'utilisation d'une ACP en pré-traitement a permis d'améliorer la précision tout en réduisant le coût de calcul. [7] et [27] appliquent l'ACP pour réduire la dimension des données d'entrée fournies à un MLP, dans le cadre de problèmes de régression. Dans certains cas, l'utilisation des composantes principales en entrée permet d'améliorer et d'accélérer l'apprentissage d'un réseau en réduisant la dimension avec une perte d'information minimale. Il s'agit d'une part de vérifier l'efficacité de la décomposition en composantes principales dans ce cas d'application, et d'autre part de la paramétrer de manière optimale. Il faut en effet choisir le nombre de composantes retenues. Celui-ci peut être choisi manuellement, ou en fonction du taux de variabilité à conserver avec la réduction de dimension. Chaque composante z_k contribue avec un taux $z_k / \sum_{k=1}^{m} z_k$ à la variabilité totale. On peut par exemple souhaiter exprimer plus de 99% de la variabilité des données, ce qui revient à choisir comme nombre de composantes le plus petit l tel que $\frac{\sum_{k=1}^{l} z_k}{\sum_{k=1}^{m} z_k}$ soit supérieur à 99%. Dans le cas du site de Cannes, choisir un seuil à 99.999% nous amène à garder seulement 39 variables en prenant la grille d'entrée la moins large (taille 150×150), ce qui entraîne une baisse notable du nombre de variables. Dans le souci de choisir des paramètres peu dépendants des données d'entraînement, il est cependant préférable de fixer un nombre préalable de composantes à garder. En effet, en choisissant un seuil de variabilité à conserver, la dimension d'entrée retenue peut varier simplement en changeant le découpage des données en jeux d'entraînement/validation. La recherche des meilleurs hyperparamètres du modèle se basera donc sur trois seuils de compression : $50, 100$ et 125 composantes principales.

3.2.2 Auto-encodeur

Les auto-encodeurs [29] permettent également d'effectuer une réduction de dimension, en se basant cette fois sur les réseaux de neurones. Un auto-encodeur basique est un réseau de neurones présentant le même type d'architecture qu'un réseau de type MLP, avec la contrainte que les dimensions d'entrée et de sortie doivent être identiques. Les couches intermédiaires présentent un nombre de neurones inférieur à la dimension d'entrée, dans le but de réduire cette dimension et de compresser les données. Le réseau apprend à projeter les données dans un espace de dimension réduite, puis à les restituer pour retrouver les données qui lui sont fournies en entrée. Les données d'entrée sont encodées au sein des couches cachées, puis décodées en sortie. L'idée est donc que le résultat de l'encodage donne une version compressée des données, avec une dimension réduite mais en gardant une quantité d'information suffisante pour les reconstituer. Outre la réduction de dimension, différents modèles d'auto-encodeurs sont notamment utilisés pour le débruitage [17], l'extraction des caractéristiques d'une image [19] ou encore la génération d'image [10].

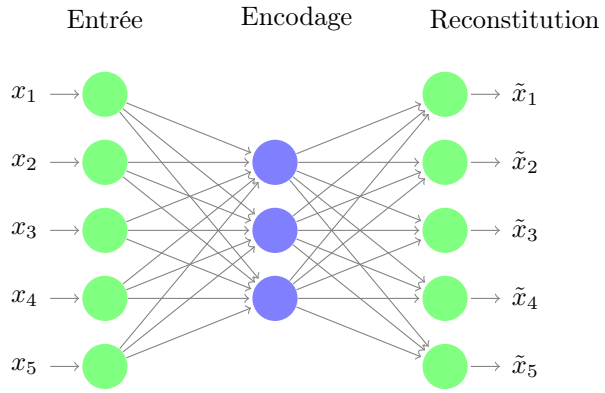
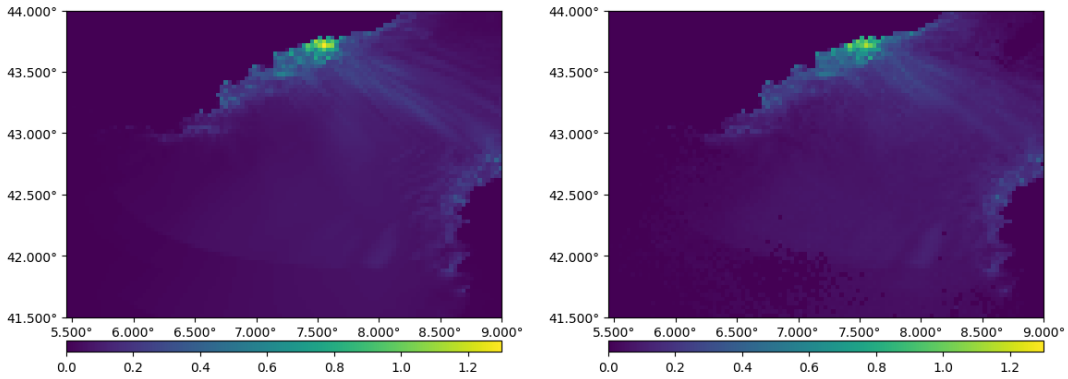
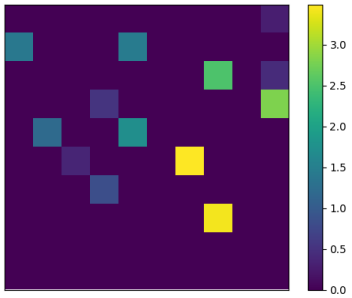


FIGURE 7 – Architecture d'un auto-encodeur à une couche cachée



(a) Grille mère originelle

(b) Grille mère reconstituée



(c) Grille encodée

FIGURE 8 – Codage/décodage de la grille mère par auto-encodeur

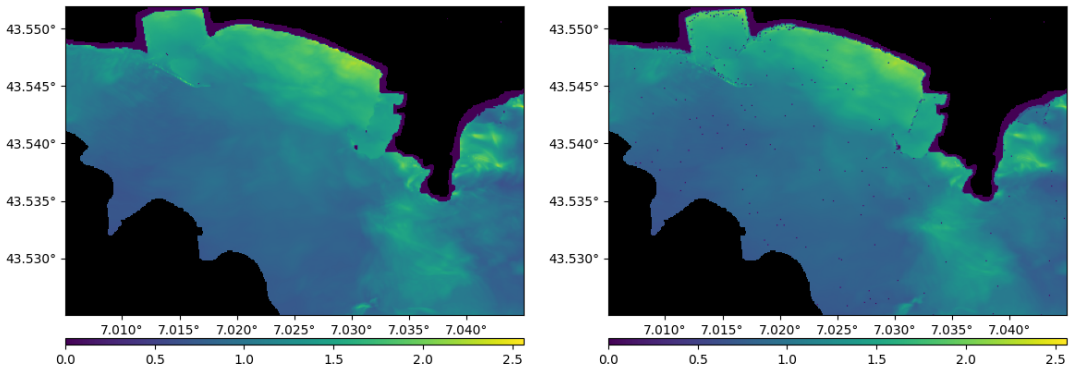
On retrouve un processus similaire dans le modèle MLP basique, les corrélations entre les données d'entrée et de sortie étant compressées dans les couches cachées lors de l'apprentissage. Cependant l'auto-encodeur se concentre spécifiquement sur l'encodage des données d'entrée, on peut donc espérer que la représentation extraite de la couche d'encodage contiendra uniquement l'information nécessaire aux prédictions et permettra d'améliorer l'entraînement du réseau de prédiction.

Cette méthode de réduction de dimension présente toutefois plusieurs désavantages directs par rapport à l'analyse en composantes principale. La première difficulté est le paramétrage. Dans le cas de l'ACP, il s'agit simplement de choisir le nombre de composantes gardées, en sachant qu'un nombre élevé de composantes permet de restituer très efficacement les données. L'auto-encodeur est un réseau de neurones dont il faut sélectionner les hyperparamètres (nombre de couches cachées, de neurones, ...). L'entraînement du réseau est également un problème, particulièrement dans le cas de Cannes, le nombre de scénarios étant relativement faible, le risque de sur-apprentissage est par conséquent élevé. Sélectionner un taux de *dropout* fort gêne ici l'apprentissage et bruite le résultat. Rajouter une couche cachée avant et après l'encodage aggrave l'*overfitting*, une architecture simple à une couche d'entrée, une couche cachée d'encodage et une couche de sortie est donc privilégiée.

3.3 Régularisation

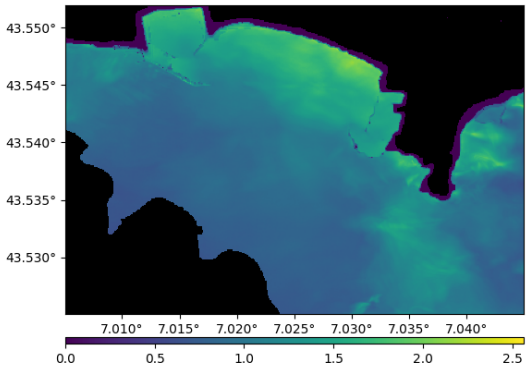
Différentes méthodes de régularisation sont essayées ici pour limiter les phénomènes d'*overfitting*.

La première technique utilisée est celle du *dropout*. Le principe du *dropout* est de couper aléatoirement une partie des sorties d'une couche cachée durant l'apprentissage. Le *dropout* agit comme une forme de bruit à l'intérieur du réseau. Le taux de *dropout* préconisé à 0.5 selon l'article originel [9] a une incidence importante sur l'efficacité de l'opération. Un taux trop faible est inefficace, le réseau parvenant à ignorer le bruit. Un taux trop important peut détériorer l'apprentissage voir le rendre inepte. Le taux idéal étant sensible au problème, on compare un taux de *dropout* faible et un taux fort. On tente également d'ajouter une contrainte de régularisation L^2 sur les paramètres des couches cachées. Un terme additionnel de pénalisation contrôlant

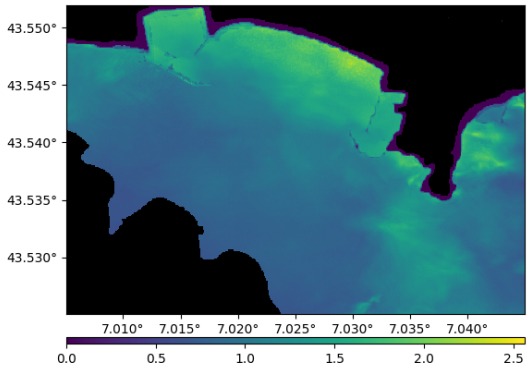


(a) Grille exacte

(b) Sans régularisation



(c) Dropout à 0.2



(d) Dropout à 0.2 et batch-normalization

FIGURE 9 – Grilles filles prédites pour un séisme de magnitude 6.8 en zone 5

l'amplitude des poids est ainsi rajouté à la fonction coût :

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_j \theta_j^2 \tag{15}$$

Les θ_j désignent l'ensemble des poids du modèle et λ une constante de pénalisation, fixée ici à 10^{-3}.

Enfin, on essaye également de rajouter une étape de *batch-normalization*. Celle-ci consiste à centrer-réduire la sortie d'une couche par *batch*. Durant l'entraînement, si l'on note x et y les entrées et sorties d'une couche de *batch-normalization* et μ, σ^2 les espérance et variance du *batch* contenant x on a la relation :

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{16}$$

Ici ϵ est une constante de lissage, et γ, β deux paramètres de normalisation qui sont optimisés par rétropropagation du gradient durant l'entraînement. Au moment de faire une prédiction, l'opération 16 est répétée en remplaçant μ et σ^2 par une moyenne et une variance calculées et mises à jour à chaque *batch* vu durant l'entraînement.

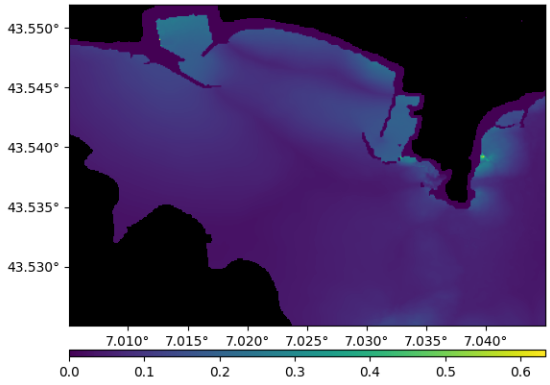
Le phénomène d'*overfitting* est visible à l'oeil nu en absence de régularisation : la figure 9 compare les prédictions obtenues avec et sans *dropout*. Certains pixels deviennent noirs en absence de régularisation, pour différentes entrées. Les sorties correspondantes sont négatives et sont donc annulées par la fonction d'activation de la couche sortante. Les pixels touchés varient d'un entraînement à l'autre, selon l'initialisation aléatoire des poids et le déroulement de l'entraînement. Le modèle utilisé ici contient deux couches cachées et 256 neurones par couche, avec un taux d'apprentissage fixé à 5.10^{-4}, mais ce phénomène est observable quelque soit la composition de couches utilisée. L'ajout de *dropout* (et donc de bruit) perturbe cet effet.

LR	dropout	L2	Batch-Normalization	RMSE
$1 \cdot 10^{-4}$	0.0	0	Non	0.09535
$1 \cdot 10^{-4}$	0.2	0	Non	0.08769
$1 \cdot 10^{-4}$	0.5	0	Non	0.09037
$1 \cdot 10^{-4}$	0.2	$1 \cdot 10^{-3}$	Non	0.11067
$1 \cdot 10^{-4}$	0.2	0	Oui	0.09169

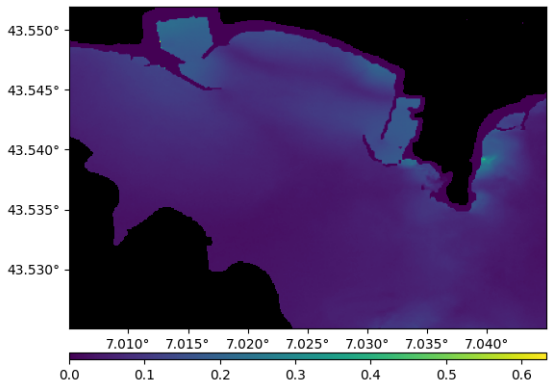
TABLE 1 – Erreurs obtenues pour différentes régularisations

Le tableau 1 compare les erreurs RMSE obtenues sur le jeu de validation pour un modèle contenant une couche cachée à 1024 neurones résultat de l'optimisation d'hyperparamètres effectuée en section 3.5.1. On fait varier le niveau de régularisation en prenant des taux de *dropout* différents et en ajoutant de la régularisation L^2 ou de la *batch-normalization*. L'erreur RMSE est prise sur l'ensemble de la grille prédite, soit entre $-50m$ et $+8m$. Les valeurs reportées dans le tableau sont issues d'une moyenne des erreurs obtenues sur 3 entraînements à 200 itérations. On note qu'appliquer un dropout faible permet de réduire légèrement l'erreur, ce qui est une conséquence de la disparition des cellules noires, mais peut également être lié à une meilleure capacité du modèle à généraliser. Utiliser une régularisation L^2 semble détériorer les résultats de manière significative. La *batch-normalisation* a peu d'effet sur l'erreur mais détériore les prédictions pour certains scénarios, en particulier à faible amplitude. La figure 10 présente un exemple de dysfonctionnement, où les hauteurs prédites sont nulles.

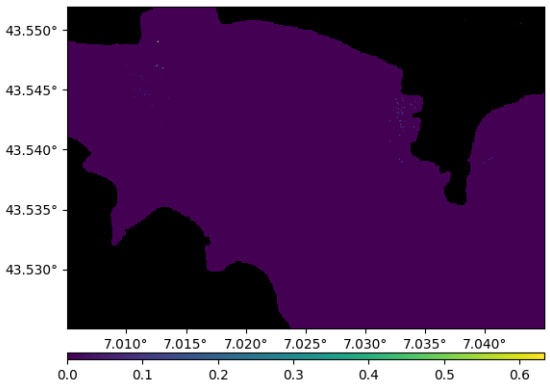
Au vu des erreurs obtenues, il semble pertinent d'appliquer uniquement un *dropout* avec un taux faible de 0.2. Les artefacts observés en absence de régularisation pourraient être éliminés en post-traitement, par exemple en moyennant les valeurs autour des valeurs nulles isolées. L'utilisation d'une régularisation, même faible, permet cependant d'atténuer les effets du surapprentissage qui peut réduire la capacité du modèle à se généraliser à de nouvelles données.



(a) Grille exacte



(b) Dropout à 0.2



(c) Dropout à 0.2 et batch-normalization

FIGURE 10 – Grilles filles prédites pour un séisme de magnitude 6.8 en zone 4

3.4 Ajout des paramètres de source et normalisation

L'une des difficultés présentées au modèle est de reconstituer une image haute fidélité avec assez peu d'informations sur les valeurs prises dans la zone, et dans le cas du jeu de données de Cannes une base d'entraînement limitée. Une manière d'augmenter la quantité de données fournies en entrée sans effectuer de simulations supplémentaires est d'ajouter les paramètres de source en entrée du modèle. Ces paramètres servent à l'initialisation des conditions initiales pour les simulations produites par le code de calcul. On retient parmi ces paramètres les coordonnées de la source, la magnitude du séisme et l'angle d'orientation de la faille, qui sont extraits des fichiers de configuration des simulations. Ces informations supplémentaires peuvent fournir un aperçu de l'origine et de la direction des tsunamis considérés.

Dans le cadre d'un modèle MLP, il est possible de simplement rajouter ces données au vecteur des entrées. Cependant, les écarts de valeurs entre les différentes variables appartenant à des catégories différentes devenant importants, il peut être intéressant d'effectuer un pré-traitement. Des études [13] ont montré l'impact positif que pouvait avoir la normalisation des données d'entrée sur les performances d'un réseau de neurones. L'une des normalisations standards est le *min-max scaling*. Le principe est de ramener toutes les variables à l'intervalle $[0; 1]$. Ceci nécessite de déterminer les bornes de chaque variable. Dans le cas de la hauteur de vague, on pourrait poser une hauteur limite qui n'est jamais dépassée en pratique. Dans un cadre plus général où les bornes des variables sont inconnues, on peut estimer les valeurs maximale et minimale à l'aide du jeu d'entraînement et effectuer la mise à l'échelle de chaque variable :

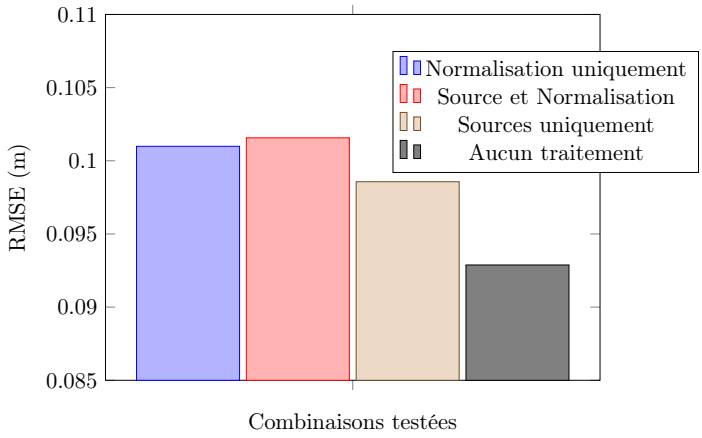
$$\frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{17}$$

Les mêmes valeurs extrémales seront ensuite utilisées pour mettre à l'échelle les jeux de test et de validation. Un premier test est de comparer les performances de quelques modèles avec et sans normalisation.

La figure 11 résume quelques résultats obtenus en ajoutant une normalisation et/ou les paramètres de source aux entrées. Les modèles testés figurent parmi les meilleurs modèles retenus lors du choix des meilleurs

Couches	Neurones	LR	Réduction	Normalisation	Source	RMSE (m)
1	1024	$1 \cdot 10^{-4}$	Non	Oui	Non	0.10099
1	1024	$1 \cdot 10^{-4}$	Non	Oui	Oui	0.10157
1	1024	$1 \cdot 10^{-4}$	Non	Non	Oui	0.09857
1	1024	$1 \cdot 10^{-4}$	Non	Non	Non	0.09288
1	1024	$2.27403 \cdot 10^{-4}$	100 (ACP)	Oui	Non	0.12668
1	1024	$2.27403 \cdot 10^{-4}$	100 (ACP)	Oui	Oui	0.1178
1	1024	$2.27403 \cdot 10^{-4}$	100 (ACP)	Non	Oui	0.40174
1	1024	$2.27403 \cdot 10^{-4}$	100 (ACP)	Non	Non	0.07908

(a) Erreurs obtenues



(b) Tracé des erreurs (modèle sans ACP)

FIGURE 11 – Résultats obtenus avec et sans normalisation et paramètres de source

paramètres (voir la section 3.6). La normalisation des données d'entrée semble ici dégrader la qualité de l'apprentissage, que ce soit en prenant une borne maximale fixée à 10 ou en prenant le maximum obtenu sur le jeu d'entraînement pour chaque variable. Les erreurs sont calculées sur le jeu de validation uniquement. L'ajout des paramètres de source au vecteur d'entrée, avec ou sans normalisation, ne permet pas d'obtenir des performances semblables à celles obtenues en utilisant uniquement la grille mère et a tendance à perturber l'apprentissage. L'essai de différentes combinaisons de paramètres de source (non présentes dans le tableau) a donné des résultats similaires.

3.5 Choix des hyperparamètres du modèle

Bien que le modèle considéré ici soit particulièrement simple, plusieurs hyperparamètres doivent tout de même être ajustés. Les principaux paramètres ajustables sont bien sûr le nombre de couches et de neurones par couche. Le taux d'apprentissage (*learning rate*) est également un paramètre modifiable. Quelques essais pour différents nombres de couches et de neurones semblent indiquer une bonne performance pour une architecture peu profonde (peu de couches cachées avec un bon nombre de neurones). Choisir un nombre de couches plus important ne semble pas à priori améliorer de façon significative les résultats.

Il est néanmoins intéressant d'explorer davantage de configurations et de façon plus systématique afin de déterminer si un ajustement différent de cette architecture assez basique peut améliorer les résultats. La procédure de recherche a donc été la suivante : le taux de *dropout* a été fixé à 0.2 et la pénalisation L^2 a été désactivée. Les autres paramètres (profondeur, nombre de neurones, taux d'apprentissage) sont déterminés en utilisant un algorithme d'optimisation. On applique l'algorithme dans chaque cas en prenant la grille d'entrée la plus petite, et la grille de sortie standard tronquée entre $-50m$ et $+8m$ de profondeur. La métrique de comparaison entre les modèles sera simplement ici l'erreur MSE calculée entre $-50m$ et $8m$ sur le jeu de validation. On comparera ensuite les meilleurs modèles trouvés par la recherche en les évaluant sur chaque grille de sortie, et en effectuant une *cross-validation* sur la grille de sortie principale.

3.5.1 Recherche aléatoire et optimisation bayésienne

Différents algorithmes de recherche des hyperparamètres optimaux d'un modèle existent. Les plus communs sont la recherche par grille, la recherche aléatoire, l'optimisation bayésienne [24] et plus récemment l'algorithme Hyperband [16]. Ces trois derniers algorithmes sont implémentés dans la librairie *Keras-Tuner*, qui a été utilisée ici.

Dans le cadre de ce réseau, le nombre de paramètres à ajuster est faible, mais une recherche exhaustive par grille reste coûteuse en calculs. Un algorithme de recherche par optimisation bayésienne a été appliqué pour quatre niveaux de réduction de dimension (aucune réduction, 50, 100, 125 composantes principales). La grille de sortie utilisée est la grille 1 de Cannes (figure 6). Différents choix de profondeur de réseau, d'épaisseur de couche et de taux d'apprentissage sont comparés en effectuant 50 itérations de l'algorithme (et donc 50 entraînements).

L'optimisation bayésienne [24] consiste à rechercher le minimum (ou le maximum) d'une fonction f en construisant un modèle probabiliste de f. Le but est de trouver cet optimum en effectuant peu d'évaluations de f, les évaluations de f étant coûteuses. Pour cela, l'information recueillie lors de chacune des évaluations précédentes de f est utilisée pour affiner le modèle et déterminer le point d'évaluation suivant. Ceci présente un surcoût de calcul, qui reste intéressant lorsque le coût d'évaluation de f est important. Ici la fonction f à optimiser est la fonction qui à un ensemble d'hyperparamètres $x \in \mathbb{R}^d$ associe la valeur de la fonction coût évaluée sur les données de validation après l'entraînement d'un modèle construit avec ces hyperparamètres. Une évaluation de f correspond donc à un entraînement de modèle, ce qui justifie l'application de calculs supplémentaires pour réduire le nombre d'évaluations. L'approximation probabiliste de f utilisée ici s'appuie sur les processus gaussiens. L'algorithme nécessite une fonction d'acquisition, qui va déterminer le point d'évaluation suivant en se basant sur toutes les évaluations précédentes de f. La librairie *Keras-Tuner* utilise une fonction d'acquisition de type GP-LCB (*Gaussian Process - Lower Confidence Bound*) :

$$\alpha(x) = \mu(x) - \beta\sigma(x) \tag{18}$$

où μ, σ désignent respectivement l'espérance et l'écart type du modèle. L'algorithme va chercher à minimiser cette fonction pour calculer le point d'évaluation suivant. Le premier terme de la formule 18 est un

Paramètre	Valeurs possibles
Nombre de couches	2, 3, ..., 6
Nombre de neurones	32 à 1024, pas de 64
Taux d'apprentissage	10^{-4} à 10^{-2}, échelle logarithmique

TABLE 2 – Domaine de recherche des meilleurs hyperparamètres

terme d'exploitation : on recherche un point qui minimise la fonction f. Le second terme est un terme d'exploration : on met la priorité sur les régions peu explorées, pour lesquelles la variance du modèle représentant f est forte. Plus le paramètre β (fixé à l'avance) est grand, plus la fonction d'acquisition mettra la priorité sur l'exploration et testera des combinaisons de paramètres inconnues.

Des bornes limites assez hautes ont été retenues pour la profondeur et le nombre de neurones, au risque de trouver des réseaux optimaux très complexes. Le domaine de recherche est résumé en tableau 2. Les réseaux obtenant les meilleures performances au cours de la recherche seront étudiés pour déterminer s'ils justifient leur complexité par des performances significativement meilleures, sans *overfitting* visible. Les critères de comparaison seront détaillés en section 3.6. La recherche s'effectue en limitant l'entraînement de chaque modèle à 300 itérations avec un critère d'arrêt anticipé (*early stopping*) en cas de stagnation de l'erreur quadratique moyenne après 30 itérations sur le jeu de validation. Ce critère d'arrêt permet de limiter le sur-apprentissage et d'éviter de prolonger la durée d'entraînement inutilement, le nombre d'entraînement à effectuer étant important (50 dans le cadre de cette recherche). La meilleure erreur de validation obtenue au cours de l'apprentissage est retenue et sert de score de comparaison entre les modèles. On obtient des résultats assez similaires pour les différents niveaux de réduction de dimension : les modèles obtenant les meilleures performances contiennent peu de couches cachées, un grand nombre de neurones par couche et un taux d'apprentissage faible. Les deux meilleurs résultats obtenus pour chaque recherche sont résumés en annexe dans le tableau 9.

3.5.2 Taux d'apprentissage adaptatif

On peut finalement essayer d'améliorer les résultats obtenus en choisissant un taux d'apprentissage adaptatif. On choisit ainsi un taux d'apprentissage à décroissance exponentielle, défini selon la formule 19.

$$l(n) = l_0 \rho^{\frac{n}{N}} \tag{19}$$

l_0 est ici le taux initial, n désigne l'étape d'apprentissage (chaque passage d'un *batch* constitue une étape) et N le nombre d'étapes avant une mise à jour du taux ($\frac{n}{N}$ désigne ici une division entière). n ne doit pas être confondu avec le nombre d'itérations (ou *epochs*) qui correspondent chacune à un passage sur l'ensemble

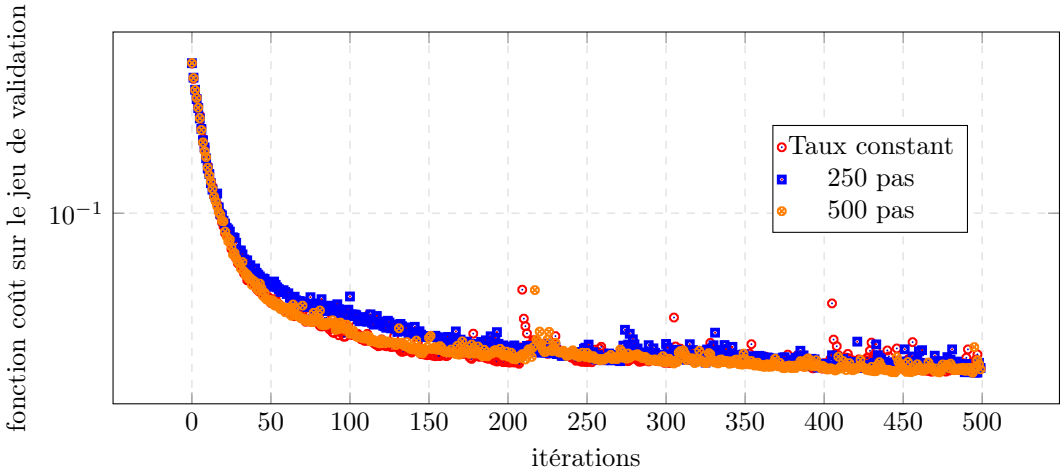


FIGURE 12 – Fonction coût évaluée sur la validation en fonction des itérations pour différentes vitesses de décroissance

des *batches* du jeu d'entraînement. Le coefficient ρ est fixé à 0.94. Différents pas de décroissance sont testés, le but étant de trouver une vitesse de décroissance suffisante pour pouvoir dépasser le plateau rencontré en fin d'entraînement. Une vitesse de décroissance trop importante peut cependant avoir un effet négatif sur l'apprentissage, un taux d'apprentissage trop faible pouvant entraîner une stagnation en début d'entraînement ou un fort ralentissement de l'apprentissage.

La figure 12 présente l'évolution de la fonction coût pour plusieurs entraînements du même modèle, sans diminution du taux d'apprentissage, et avec une diminution toutes les 25 ou 50 itérations (respectivement 250 et 500 pas d'entraînement pour Cannes).

La diminution du taux d'apprentissage ne semble pas apporter de gain significatif. On observe toujours une stagnation assez rapide de l'erreur de validation, sans gain notable. La seule différence observable sur les courbes d'entraînement est une diminution de la variabilité lorsque le nombre d'itération est grand, le facteur utilisé dans la descente de gradient étant inférieur en temps long. Cependant, pour un tel nombre d'itérations, l'erreur stagne déjà et le modèle est en situation de surapprentissage.

3.5.3 *Batch size*

La taille de *batch* ne semble avoir ici qu'un impact mineur sur les résultats de l'entraînement. La figure 13 illustre l'évolution du coût en fonction des itérations pour un modèle à une couche possédant 1024 neurones. On observe une convergence en un nombre réduit d'itérations lorsque la taille de *batch* est faible, au prix d'une variance du coût plus élevée. Cette variance peut réduire le risque de blocage au niveau d'un minimum local. Conserver les poids donnant la plus petite erreur de validation permet d'éviter une augmentation significative de l'erreur à cause d'une distribution malheureuse des *batches* sur les dernières itérations d'entraînement. Le temps de calcul est similaire pour un calcul sur processeur.

L'efficacité des *mini-batches* a été montrée dans certaines études [18]. Les tailles de *batch* importantes étant en général réservées à des ensembles de données suffisamment volumineux, on fixe ici une taille de *batch* de 32 (soit une décomposition de chaque itération en dix étapes d'entraînement pour la ville de Cannes).

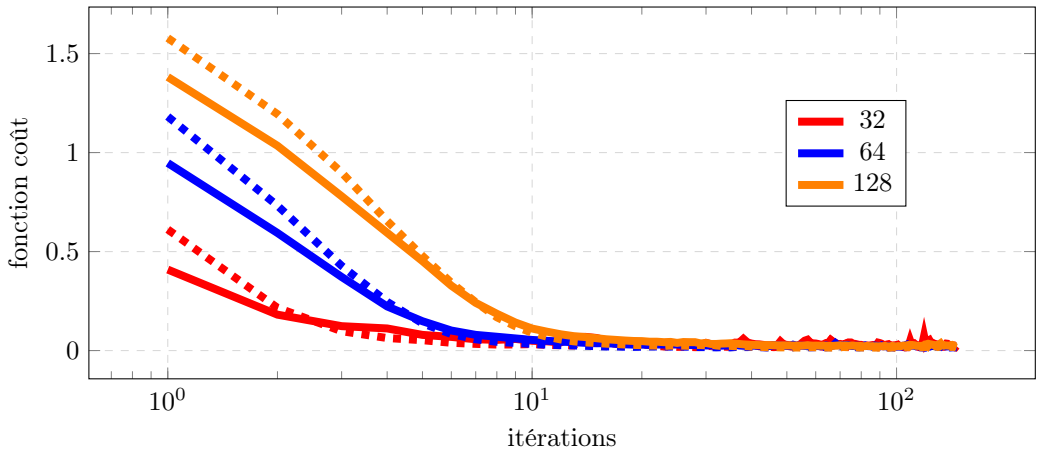


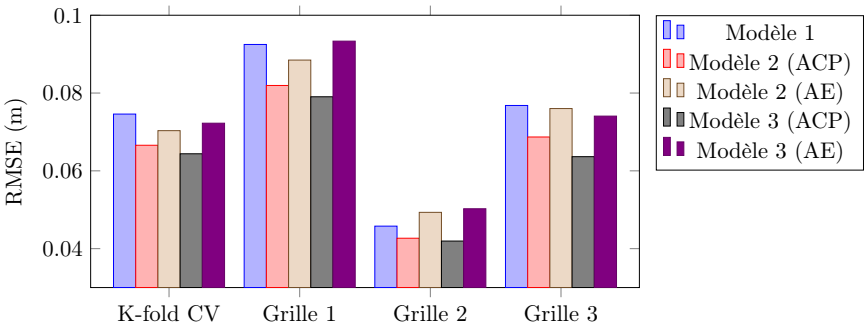
FIGURE 13 – Fonction coût sur la validation (traits pleins) et l'entraînement (pointillés) en fonction des itérations pour différentes tailles de *batch*

3.6 Sélection d'un modèle

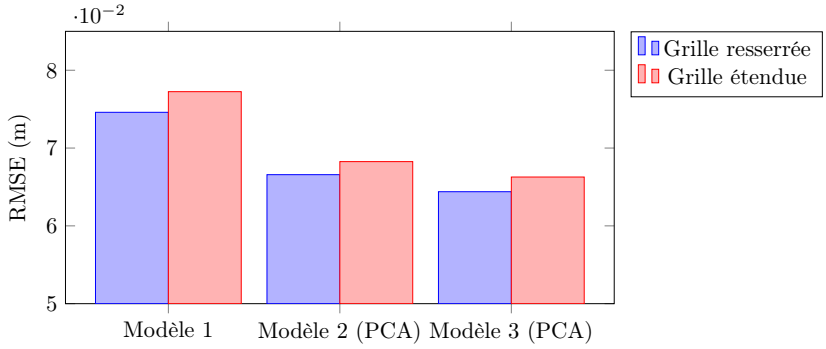
Après avoir effectué une sélection parmi les meilleures erreurs obtenues dans la recherche d'hyperparamètres, on procède à une comparaison des performances des modèles retenus. Trois architectures de modèle ont été retenues pour chacun des niveaux de réduction de dimension : aucune, puis 50, 100 et 125 composantes principales. Les modèles gardés ont été sélectionnés parmi les dix meilleurs résultats d'une recherche bayésienne. Les paramètres trouvés pour les modèles utilisant l'ACP sont également testés en effectuant une réduction de dimension par auto-encodeur (noté AE). Il s'agit ensuite de comparer tous ces modèles de manière plus poussée.

Le premier critère de comparaison se base sur une méthode de *stratified k-fold cross-validation*. Le concept est simple : le jeu de données (entraînement et validation) est sectionné en k sous ensembles (*folds*). Le modèle est entraîné k fois, en utilisant à chaque fois l'un des sous-ensembles comme ensemble de validation et le reste pour l'entraînement. On peut ensuite calculer la moyenne des erreurs obtenues à chaque entraînement. Les sous-ensembles sont choisis de sorte à respecter les proportions de chaque catégorie du jeu de données (stratification). Cette technique de validation d'un modèle est plus fiable que la simple décomposition entraînement/validation, mais est coûteuse dans le cadre d'une recherche exhaustive : en prenant $k = 6$, les 50 itérations de la recherche par optimisation bayésienne auraient demandé 300 entraînements. L'erreur obtenue est cependant nettement plus fiable que celle trouvée lors de la recherche, ne serait-ce que par le fait qu'elle est le résultat d'une moyenne sur six entraînements. L'ensemble du jeu de données est couvert (hors jeu de test), ce qui garantit une évaluation sur des données plus variées. L'erreur obtenue risque également d'être inférieure à celle observée en utilisant la décomposition entraînement/validation classique, la base d'entraînement étant plus étendue.

En parallèle de ce calcul, qui est effectué en utilisant les grilles standard d'entrée et de sortie, on récupère également les erreurs de validation obtenues en effectuant un entraînement sur les deux autres grilles de sortie. Pour limiter le nombre d'entraînements nécessaires, on utilise alors une décomposition entraînement/validation simple. Effectuer une moyenne sur plusieurs entraînements permettrait d'avoir des résultats plus précis, particulièrement lorsque l'écart entre deux modèles est faible. Les résultats obtenus sont résumés dans le tableau 10. Le tableau 11 contient les résultats obtenus par *cross-validation* en prenant la grille d'entrée large. On compare à chaque fois les résultats à un modèle standard à 2 couches et 512 neurones par couche. La figure 14 résume ces résultats graphiquement en gardant seulement les meilleurs modèles pour chaque niveau de réduction.



(a) Résultats sur pour la grille mère resserrée



(b) Erreurs de cross-validation pour les grilles mères resserrée et étendue

Modèle	Couches	Neurones	LR	Dimension d'entrée
Modèle 1	1	1024	$1 \cdot 10^{-4}$	Inchangée
Modèle 2	2	704	$4.64983 \cdot 10^{-4}$	50
Modèle 3	1	1024	$2.27403 \cdot 10^{-4}$	100

(c) Modèles considérés

FIGURE 14 – Comparaison des RMSE des modèles sélectionnés (ACP : analyse en composantes principales, AE : auto-encodeur)

On note que les modèles les plus performants trouvés par l'algorithme ne contiennent qu'une ou deux couches cachées, ce qui explique un faible écart des modèles trouvés au modèle standard. Ceci peut être dû à la faible taille du jeu de données et à un phénomène d'*overfitting* en cas de profondeur trop importante. Les erreurs calculées sur les trois grilles de sortie présentent des amplitudes très différentes, liées aux différences d'amplitude des vagues relevées entre ces zones. On observe cependant que les écarts de performances entre les modèles semblent être globalement préservés d'une zone à l'autre. L'extension de la grille d'entrée semble avoir peu d'effet, voire détériore légèrement les résultats (figure 14b).

L'utilisation d'un auto-encodeur en entrée ne semble pas améliorer les résultats, et cause même une augmentation de l'erreur pour certains modèles (figure 14a et tableau 10). Ce comportement peut s'expliquer par plusieurs facteurs. Le principal facteur d'influence tient à la composition et à l'entraînement des réseaux auto-encodeurs eux mêmes. L'architecture (nombre de couches cachées et de neurones) des auto-encodeurs a été ajustée à la main à la suite de quelques tests, et une recherche d'hyperparamètres pourrait être effectuée pour affiner ces choix. De plus, les hyperparamètres choisis pour le réseau de prédiction à chaque niveau de réduction de dimension ont été obtenus en utilisant l'analyse en composantes principale comme méthode de réduction. Effectuer une nouvelle recherche en utilisant un auto-encodeur comme méthode de réduction pourrait ainsi modifier les paramètres trouvés. Il est également probable qu'un phénomène de surapprentissage soit présent au niveau des auto-encodeurs entraînés et détériore les prédictions sur les grilles filles. Enfin, différentes types d'auto-encodeurs pourraient être envisagés, à commencer par des auto-encodeurs convolutionnels.

On pourrait également questionner l'utilisation de l'*early-stopping* : si la condition d'arrêt utilisée pour obtenir ces résultats est trop brutale, certains modèles pourraient être avantagés au détriment d'autres dont l'apprentissage se ralentirait. Un nouveau calcul de l'erreur de *cross-validation* sans arrêt anticipé mais en gardant les poids donnant la meilleure erreur de validation montre que la condition d'arrêt ne semble pas avoir d'influence sur les résultats obtenus ici.

Les meilleurs résultats (voir le tableau 9) sont obtenus en effectuant une ACP et en conservant 100 ou 125 composantes principales. Les différences entre 100 et 125 composantes étant faibles, on sélectionne un modèle prenant 100 composantes en entrée, possédant une unique couche cachée à 1024 neurones et avec un taux d'apprentissage de l'ordre de 2.10^{-4}. La partie suivante propose une évaluation plus approfondie de ce modèle.

3.7 Première évaluation du modèle retenu

3.7.1 Métriques de comparaison

La sélection précédente s'est effectuée en se basant uniquement sur l'erreur RMSE calculée sur l'ensemble de la grille de sortie tronquée prédite, soit à une profondeur comprise entre $-50m$ et $+8m$. Cependant, même dans les scénarios les plus extrêmes les points situés à une hauteur de $8m$ ont une probabilité presque nulle d'être inondés. Une partie des points va donc correspondre à une erreur nulle, qui va faire baisser l'erreur moyenne. De plus, il peut être intéressant de considérer uniquement l'erreur pour les points très proches de la côte, ou uniquement pour les inondations. On choisit ainsi de calculer trois RMSE distinctes : pour les points situés entre -50 et $+2m$, pour les points entre $-2m$ et $0m$ et pour les points entre $0m$ et $+2m$. La première erreur renseigne sur la capacité globale du modèle à effectuer des prédictions sur la grille, y compris en profondeur. La seconde concerne uniquement les prédictions proches des côtes en excluant les inondations. La dernière se concentre sur les inondations.

On ajoute à ces erreur le calcul du score R^2 du modèle. Celui-ci est donné dans le cas uni-dimensionnel (prédiction d'une sortie scalaire) par :

$$R^2(y_i, \hat{y}_i) = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{20}$$

Les y_i désignent les réalisation de la variable y et les $\hat{y}_i$ les prédictions associées du modèle. $\bar{y}$ désigne la valeur moyenne de y, le dénominateur est donc proportionnel à la variance empirique. Ce score renseigne ainsi sur la capacité du modèle à exprimer la variance des données. Un modèle constant renvoyant toujours la valeur moyenne obtient un score de 0. Un modèle donnant de moins bonnes performances que le modèle constant obtiendra un score négatif. Le meilleur score possible est 1. Dans le cas multivarié, il existe différentes

façons d'agréger les scores obtenus pour chacune des variables de sortie. Le score considéré ici effectue une moyenne pondérée selon la variance de chaque variable :

$$R^2(y_i, \hat{y}_i) = \sum_k \frac{\sigma_k^2}{\sum_j \sigma_j^2} \left(1 - \frac{\sum_i (y_{i,k} - \hat{y}_{i,k})^2}{\sum_i (y_{i,k} - \bar{y}_k)^2}\right) \tag{21}$$

Ici $y_{i,k}$ désigne la i^{e} réalisation de la k^{ie} variable, et $\bar{y}_k$ la moyenne empirique de la k^{ie} variable. $sigma_k^2$ est la variance empirique de la k^{ie} variable. Le score correspondant à chaque variable est donc pondéré par la proportion de la variabilité totale contenue par par cette variable.

De même que l'on sépare différentes erreurs RMSE selon la profondeur, on calculera le score R^2 selon les profondeurs précisées plus haut.

3.7.2 Évaluation sur Cannes

On évalue le modèle à une couche et une ACP en entrée sur les jeux de validation et de test. Dans ces deux jeux de données, les scénarios n'ont pas été utilisés directement par le modèle durant l'entraînement. La figure 15 représente les distributions des erreurs en découpant selon les tranches de profondeur décrites précédemment pour les 74 scénarios du jeu de validation. Les rectangles représentent le découpage des données selon les quartiles et chaque marqueur correspond à un scénario de validation donné. On relève un troisième quartile à $6.2cm$ pour l'erreur RMSE proche de la côte (entre 0 et $2m$ de profondeur). Ce quartile monte à plus de $20cm$ lorsque seules les inondations sont considérées (altitude de 0 à $2m$). Ceci s'explique par le fait que ces erreurs sont coûteuses. Les scénarios présentant des inondations fortes sont accompagnés de hauteurs de vagues importantes, et une cellule pour laquelle aucune inondation n'est prédite à tort cause un surcoût d'erreur important. Au contraire, les hauteurs au large sont plutôt bien prédites en moyenne.

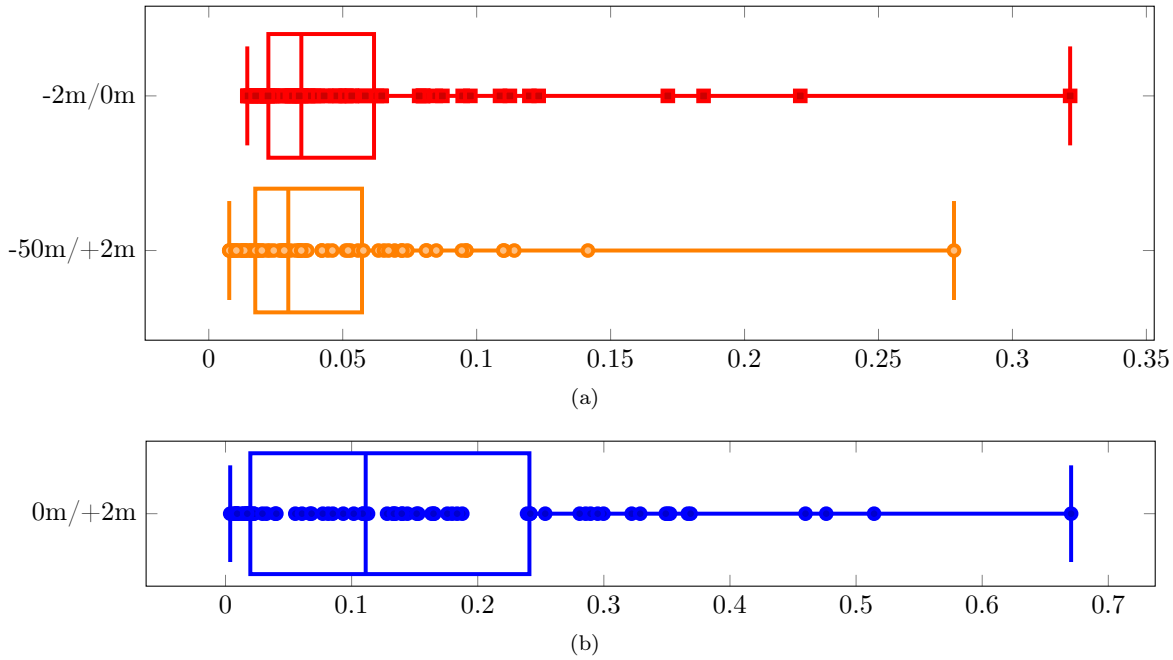
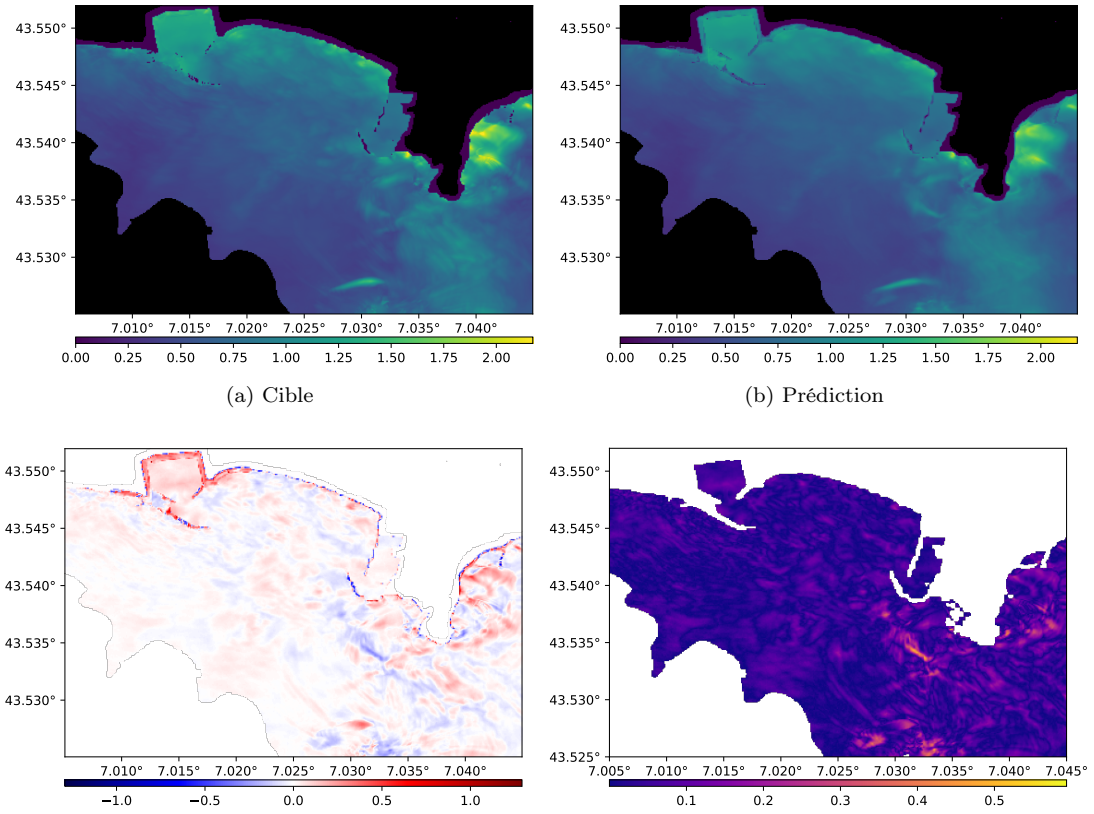


FIGURE 15 – RMSE du modèle 3 sur le jeu de validation selon la profondeur (en m)

Cependant, l'erreur sur le jeu de validation a servi à discriminer et à sélectionner les modèles, et conditionne en partie l'arrêt de l'entraînement. Il est donc plus prudent de vérifier l'erreur obtenue sur un jeu de données inutilisé : c'est le rôle du jeu test. Le tableau 3 compare les erreurs globales à différentes profondeurs sur les jeux test et de validation, ainsi que les scores R^2. On trouve des erreurs du même ordre, les fluctuations pouvant être liées à la taille réduite du jeu de test.

Jeu	RMSE (-50/+2)	RMSE (-2/0)	RMSE (0/+2)	R2 (-50/+2)	R2 (-2/0)	R2 (0/+2)
Validation	0.05962	0.0736	0.20647	0.98281	0.98562	0.89526
Test	0.05126	0.05859	0.20772	0.94842	0.96969	0.70848

TABLE 3 – Comparaison des erreur RMSE (en m) et des scores R^2



(a) Cible

(b) Prédiction

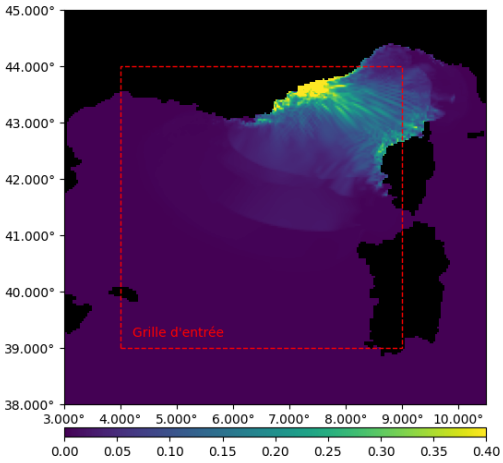
(c) Erreur signée

(d) Erreur relative en valeur absolue

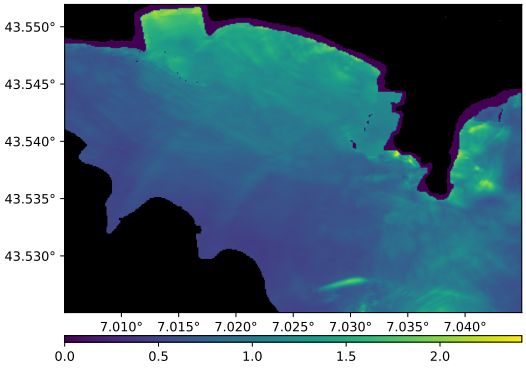
FIGURE 16 – Scénario de magnitude 7.0 en zone 5

Les figures 16 et 17 proposent les tracés d'erreur cellule par cellule pour deux scénarios à forte amplitude du jeu de test. L'erreur signée est la soustraction de la valeur prédite à la valeur exacte, une erreur négative signifie donc une surestimation et une erreur positive une sous-estimation. L'erreur relative est le quotient de la valeur absolue de cette différence par la valeur exacte, uniquement en les points situés sous le niveau de la mer pour éviter les divisions par zéro.

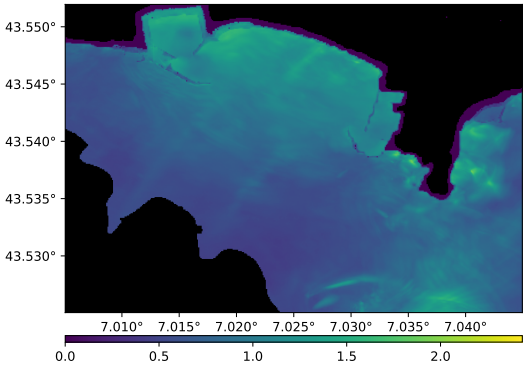
Les erreurs RMSE entre $-50m$ et $2m$ sont respectivement de $13cm$ et $8cm$. Les erreurs au niveau des inondations uniquement sont de $48cm$ et $34cm$.



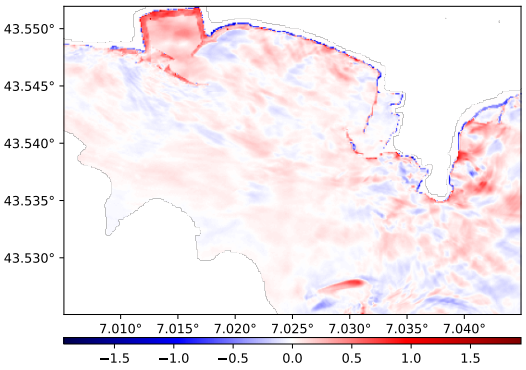
(a) Hauteur maximale d'entrée (m)



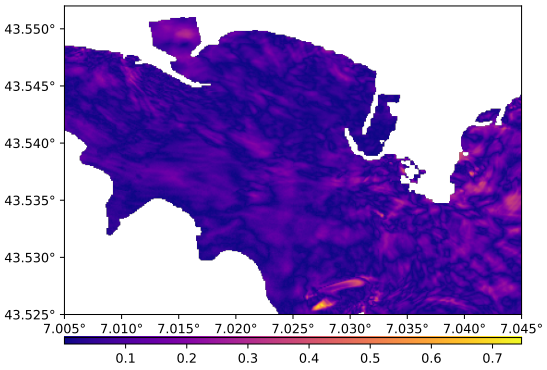
(b) Hauteur maximale cible (m)



(c) Hauteur maximale prédite (m)



(d) Erreur signée (m)



(e) Erreur relative en valeur absolue

FIGURE 17 – Hauteur de vague pour un scénario de magnitude 7.2 en zone 5

4 Mise en place de techniques d'évaluation

L'entraînement de différents types de modèles a montré la difficulté à comparer et à évaluer la qualité des prédictions de ces modèles. En particulier nous avons vu qu'une erreur statistique simple comme la RMSE prise sur l'ensemble de la grille donne seulement un aperçu partiel des capacités d'un modèle, qui peut être difficilement interprétable. Cette section vise à développer différentes méthodes d'évaluation des modèles et de représentation de l'erreur commise dans les prédictions. La création du second jeu de données sur Nice permettra d'évaluer les modèles sélectionnés sur un nouveau cas d'étude proposant des scénarios plus variés.

4.1 Segmentation des données de validation

Bien que l'on souhaite obtenir une adéquation optimale entre les prédictions du modèle et les simulations en chaque point, il est avant tout nécessaire, en contexte d'alerte, que le modèle soit capable de distinguer un scénario dangereux présentant des hauteurs de vague importantes d'un scénario à très faible amplitude. De plus, il est difficile de comparer l'erreur commise entre deux scénarios proposant des amplitudes très différentes. L'erreur relative est souvent difficile à interpréter, particulièrement lorsque les variations d'amplitude des valeurs considérées sont importantes. En effet, une erreur de 20% commise sur une hauteur de $2cm$ est en pratique négligeable dans ce cas d'application, tandis que la même erreur relative sur une hauteur de $2m$ ne l'est pas.

Enfin, les scénarios les plus extrêmes sont cruciaux à l'évaluation des modèles puisqu'ils représentent un risque important. La problématique est qu'ils sont à la fois rares et d'une amplitude très supérieure aux scénarios usuels, ce qui rend encore une fois l'évaluation globale difficile.

On peut ainsi choisir de diviser le jeu de validation en différentes catégories de scénarios "proches", puis d'évaluer ces catégories séparément. Le principal facteur agissant sur l'erreur étant l'amplitude des vagues, on choisit de segmenter les données en trois sous-ensembles correspondant aux basses, moyennes et fortes amplitudes. L'amplitude est mesurée en prenant la moyenne des hauteurs maximales entre une profondeur de deux mètres et la côte. La séparation entre les trois sous-ensembles est alors définie par la médiane et le quantile d'ordre 0.9. La moitié des scénarios constitue donc les scénarios à basses amplitudes et les dix pourcents supérieurs les scénarios à hautes amplitudes. Cette division est un choix arbitraire et présente bien sûr des limites, particulièrement si l'on considère l'évaluation des scénarios situés à la démarcation entre deux catégories.

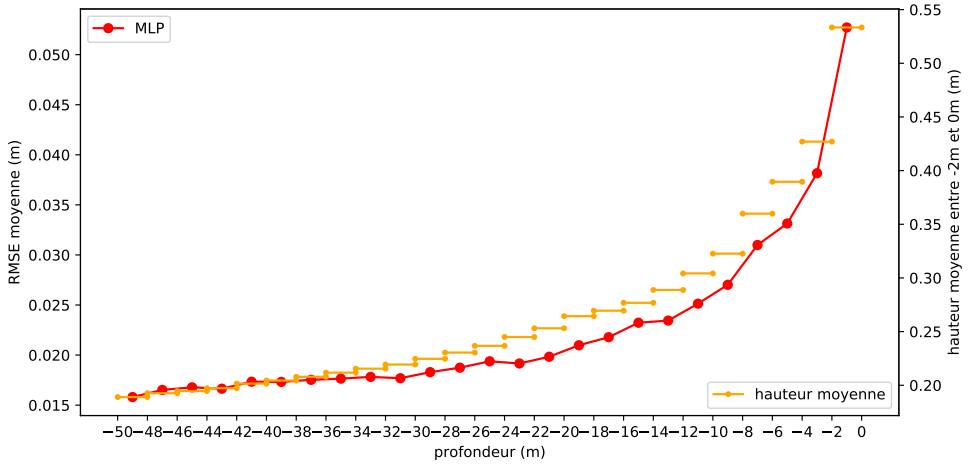
4.2 Erreur selon la profondeur

L'observation des grilles d'erreurs indique souvent une concentration de l'erreur au plus proche des côtes. De plus, la hauteur maximale des vagues va avoir tendance à augmenter lorsque la profondeur diminue. Il est ainsi difficile de comparer l'erreur en un point situé à la côte et un point pour lequel la bathymétrie atteint une profondeur de cinquante mètres, les hauteurs maximales de vagues étant de manière générale très différentes.

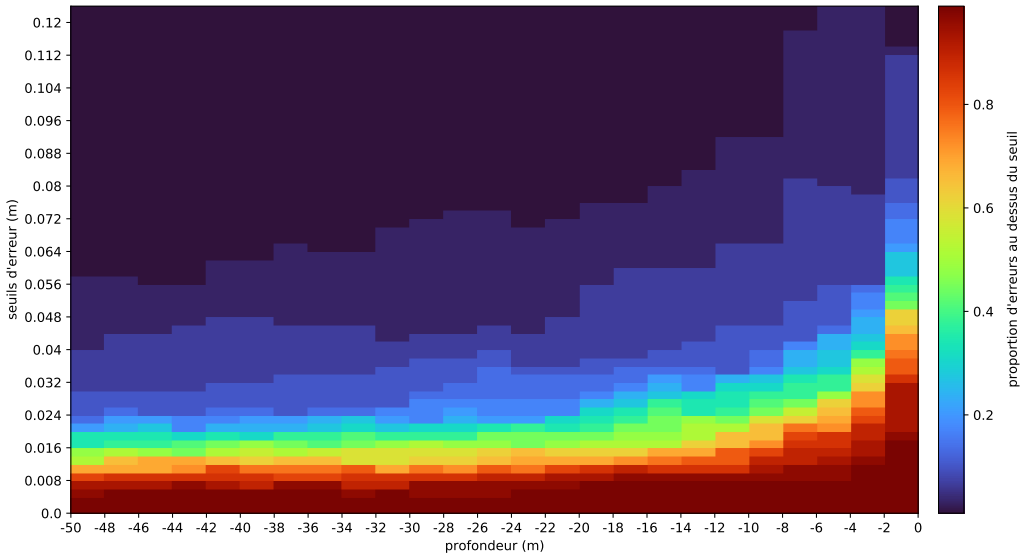
On se propose ainsi de calculer l'erreur par tranches de profondeur (ou d'altitude pour les inondations). La figure 18a présente un tel tracé d'erreur pour le MLP sur Cannes. On a calculé ici la moyenne des RMSE sur le jeu de validation en segmentant l'intervalle $[-50, 0]$ en sous-intervalles de longueur $2m$. On trace également la hauteur moyenne pour chaque tranche de profondeur pour obtenir un point de comparaison, l'échelle étant donnée par l'axe de droite.

La figure 18b propose une autre manière d'agréger les résultats trouvés pour les différents scénarios. L'échelle de couleur donne la proportion de scénarios présentant une erreur supérieure à la valeur donnée par l'axe des ordonnées, pour chaque tranche de profondeur. On obtient ainsi un meilleur aperçu de la distribution des erreurs dans le jeu de données observé.

Cette manière de calculer et de représenter l'erreur offre une meilleure vision des performances d'un modèle et permet de comparer deux modèles de manière plus efficace. Cependant, cette méthode est également très dépendante de la bathymétrie sur la grille fille, ce qui rend la comparaison des performances sur des grilles différentes complexe.



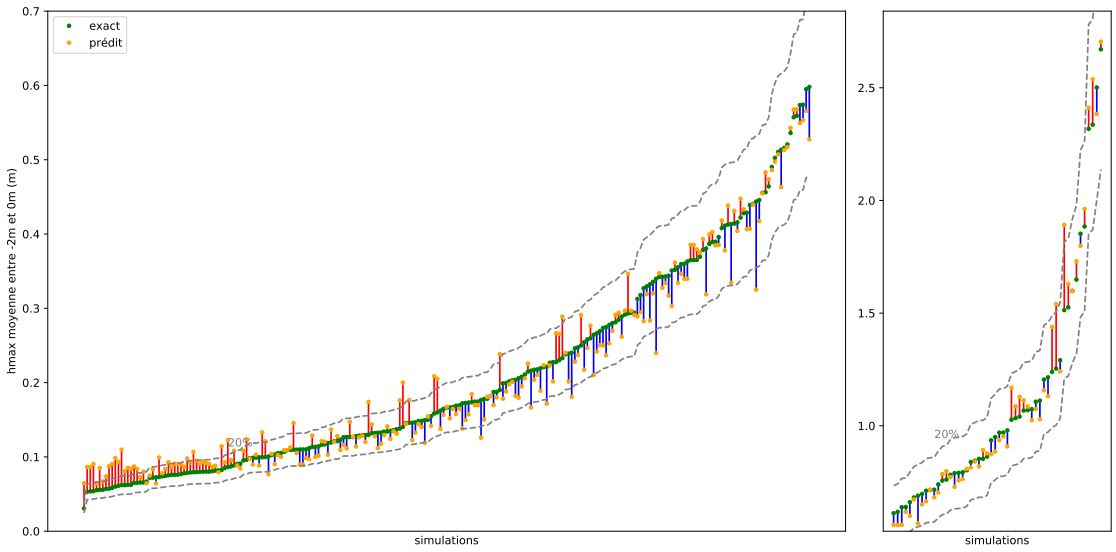
(a) Erreur moyenne par tranche de profondeur



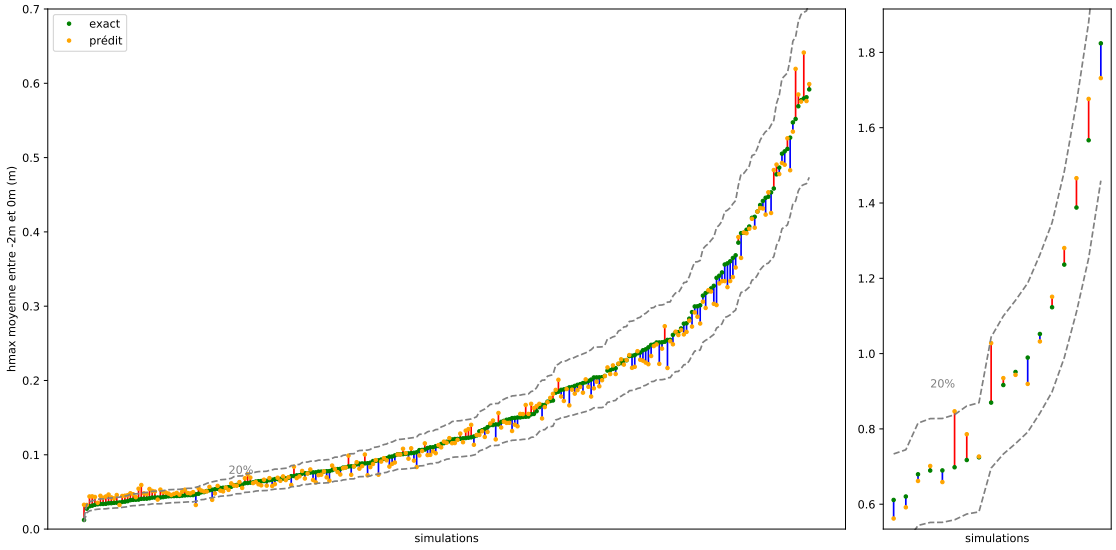
(b) Distribution des RMSE par tranche de profondeur

FIGURE 18 – Erreurs selon la profondeur pour un MLP sur Cannes (scénarios de moyennes amplitudes)

4.3 Évaluation de l'amplitude moyenne



(a) Grille 5



(b) Grille 7

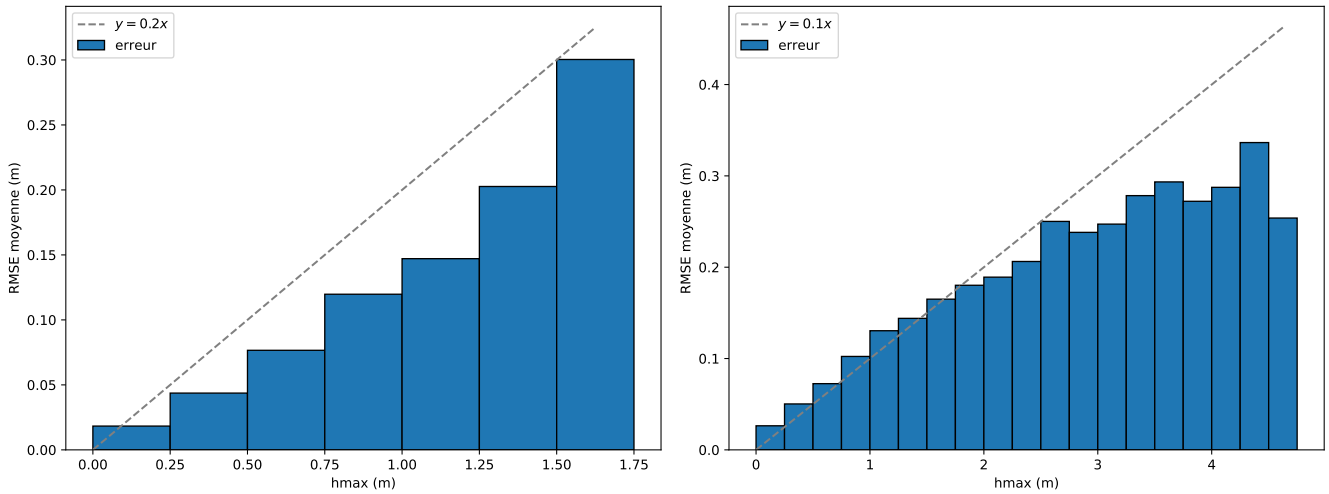
FIGURE 19 – Comparaison des hauteurs maximales moyennes entre $-2m$ et $0m$ pour un MLP sur Nice

Les tracés d'erreurs précédents permettent d'obtenir une idée du comportement moyen d'un modèle à différentes profondeurs, mais ne permet pas de visualiser les cas aberrants. De plus l'erreur ainsi agrégée est non signée (les erreurs signées pouvant se compenser dans une moyenne). On peut ainsi se ramener au calcul d'une valeur caractéristique de la grille fille et visualiser l'écart entre la valeur réelle et prédite scénario par scénario. On peut ainsi facilement visualiser l'erreur commise scénario par scénario, l'amplitude de la différence relativement à la valeur calculée et si le modèle est en sur ou sous-estimation.

Le problème est alors de trouver des valeurs caractéristiques permettant d'évaluer le fonctionnement d'un modèle, ou présentant un intérêt en contexte d'alerte. On pourrait par exemple s'intéresser au *runup*, défini généralement comme la plus haute altitude atteinte par les vagues d'un tsunami. Il est cependant peu représentatif dans ce cas : une grande partie des simulations ne présentent pas d'inondations, ou seulement de l'ordre de quelques cellules. La valeur de *run-up* correspondante n'est pas significative et est souvent imprévisible. On pourrait également considérer le nombre de cellules inondées, ce qui nécessiterait de déterminer des seuils de tolérance pour chaque bathymétrie de sortie considérée. Certaines cellules de la côte sont

inondées de manière chaotique dans les simulations, ce qui rend l'évaluation difficile. On choisit de considérer ici l'amplitude moyenne des hauteurs maximales entre $-2m$ et $0m$ de profondeur. La figure 19 présente un exemple d'une telle représentation. Un trait bleu signifie que la valeur est sous-prédite par le modèle, un trait rouge qu'elle est sur-prédite. On a tracé en pointillés la marge de 20% entourant les valeurs exactes. Bien sûr la notion d'erreur point à point disparaît de ces mesures, et l'on conserve uniquement une comparaison des valeurs moyennes des grilles exactes et prédites. Un modèle peut prédire correctement la hauteur moyenne à une certaine profondeur tout en plaçant mal les pics d'intensité. Un modèle donné peut également être plus précis qu'un autre sur une tranche de profondeur donnée mais obtenir des performances inférieures à d'autres profondeurs. On pourrait de plus regarder la hauteur moyenne sur une zone d'intérêt précise comme le quai d'un port plutôt que de se baser uniquement sur la profondeur. Enfin, on peut aussi choisir de tracer une erreur simulation par simulation pour avoir une vue d'ensemble des erreurs et pas des erreurs moyennes. Le problème reste toujours de trouver un type d'erreur représentatif des performances du modèle.

4.4 Erreur selon l'amplitude



(a) Amplitudes moyennes

(b) Amplitudes hautes

FIGURE 20 – Erreur moyenne selon la hauteur de vague maximale (grille 5 de Nice)

Une autre évaluation graphique possible est de représenter l'erreur moyenne en fonction de la hauteur de vague. La figure 20 présente un tel tracé, où chaque rectangle correspond à un intervalle de $25cm$ de hauteurs de vagues prises sur la simulation, et l'erreur obtenue est indiquée en ordonnée, moyennée sur les scénarios de validation. Le tracé correspond aux prédictions d'un MLP sur la grille 5 de Nice.

On trace également des droites permettant d'obtenir une idée de l'erreur relative correspondante. La figure 20 présente le tracé d'erreur avec un découpage selon les scénarios de moyennes et fortes amplitudes pour un MLP. On note par exemple qu'ici l'erreur reste inférieure à 10% pour les scénarios à forte amplitude, et sous les 20% pour les scénarios de moyennes amplitudes.

4.5 Évaluation d'un scénario spécifique

Les tracés précédents permettent d'obtenir un aperçu des performances des modèles sur l'ensemble des scénarios en évaluant le comportement moyen. On peut aussi vouloir évaluer un modèle individuellement sur un ou plusieurs scénarios spécifiques. Les graphiques présentés précédemment dans cette section peuvent s'adapter à ce cadre, et renseignent sur le comportement moyen d'un réseau sur une gamme de valeurs (profondeurs, amplitudes, etc.). La comparaison directe d'une simulation et de la prédiction associée peut se faire de manière plus précise. On peut notamment représenter l'erreur cellule par cellule (voir par exemple la figure 16) pour obtenir une meilleure idée de la répartition spatiale de l'erreur pour un scénario donné. On peut également représenter les hauteurs de vagues prédites en fonction des hauteurs de vagues simulées en

ayant préalablement sélectionné les points de la grille fille pour lesquels effectuer le tracé. La figure 21 propose un tel tracé pour les points situés autour de l'isobathe à $1m$ de profondeur. On peut ainsi visualiser point par point les sur ou sous-prédictions d'un modèle pour un scénario donné en mesurant l'écart par rapport à la droite d'équation $y = x$. On trace également les droites permettant de représenter une marge d'erreur de 25%. On peut ainsi facilement mettre en relation les erreurs de prédiction avec l'amplitude de la simulation aux points correspondants.

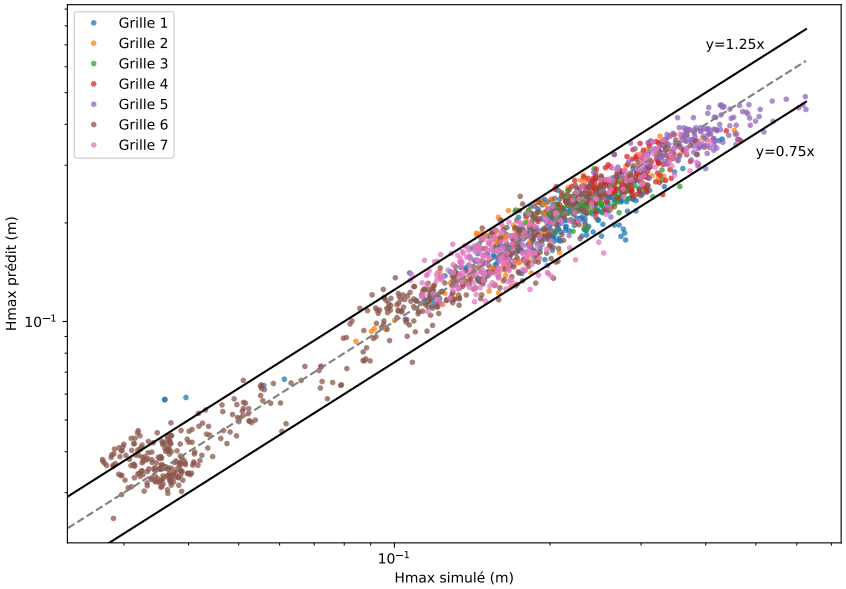


FIGURE 21 – Hauteurs prédites (m) en fonction des hauteurs simulées autour de l'isobathe $1m$ (Nice, magnitude 6.2 en zone 5)

5 Architecture à couches de convolution

Dans un second temps, nous ajoutons des couches de convolution en entrée du modèle. Le modèle obtenu peut alors être décomposé en un réseau de convolution qui génère à partir de chaque grille mère une nouvelle grille, redimensionnée ou non, à un ou plusieurs canaux, et un réseau MLP "décodeur" qui transforme cette grille (après l'avoir aplatie) pour obtenir le résultat final.

L'étape de prétraitement par ACP utilisée dans la section précédente est ainsi remplacée par une étape d'encodage par un réseau de convolution. Une restriction de cette approche est que toutes les cellules des grilles d'entrée doivent être utilisées, on ne peut plus éliminer les cellules non immergées des grilles mères avant l'entraînement.

L'un des inconvénients des réseaux de convolution profonds est l'augmentation importante du temps d'entraînement, en particulier sans l'utilisation d'accélération par GPU. La section 5.5 compare les temps de calcul des différentes architectures.

5.1 Réseau de type VGG

Le premier réseau de convolution considéré est construit selon une architecture simple de type VGG [23]. Celle-ci consiste simplement à effectuer en série des blocs de convolutions. Pour chaque bloc de convolutions successif on choisit un nombre de canaux plus important. Après chaque bloc une opération de *max-pooling* est effectuée pour réduire la taille de l'image et compresser l'information. Le réseau va donc encoder l'information de ses entrées dans une image de petite taille contenant un grand nombre de canaux. L'architecture retenue est présentée en figure 22 et comporte quatre blocs de deux couches de convolutions suivies de *max-pooling*. Chaque sortie d'une couche de convolution est représentée sur le schéma par un parallélépipède. Après chaque bloc de convolution, les dimensions de l'image sont divisées par deux tandis que le nombre de canaux (représenté par l'épaisseur des couches sur le schéma) est multiplié par deux. Une fonction d'activation RELU est appliquée à la sortie de chaque couche de convolution.

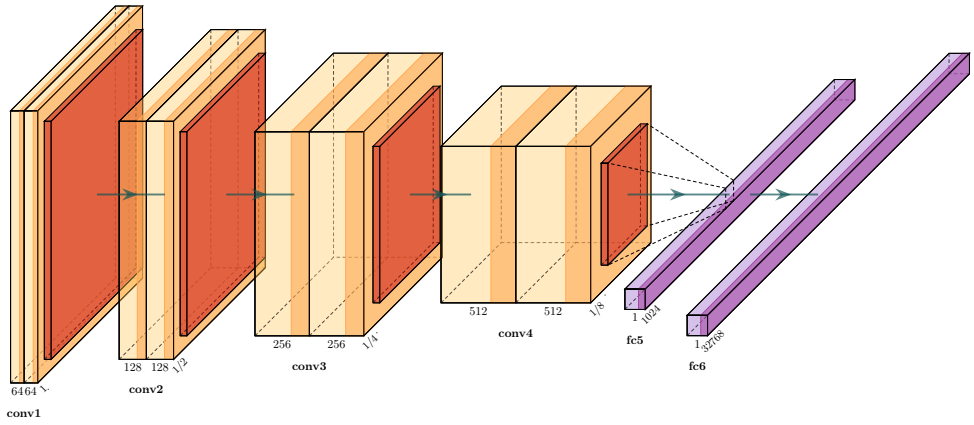


FIGURE 22 – Architecture de type VGG. Schéma généré à l'aide du code PlotNeuralNet [11]. L'épaisseur d'une couche représente le nombre de canaux.

Le résultat de la dernière phase de *max-pooling* est finalement écrasé en un vecteur qui sera traité par des couches denses. On retrouve ainsi un réseau MLP "décodeur" ayant la même structure que précédemment (une couche cachée à 1024 neurones).

Le nombre d'hyperparamètres est ici plus important que dans le cas du MLP : le nombre de blocs de convolution, le nombre de convolutions par bloc, les dimensions des noyaux, les paramètres des couches de sortie denses et les paramètres de régularisation sont autant de variables ajustables. Un premier ajustement de l'architecture est effectué manuellement par essai/erreur, en commençant à un niveau de profondeur assez important. L'ajout de *spatial-dropout* dans les couches de convolution a tendance à détériorer les performances, mais des instances de *dropout* sont conservées entre les couches denses. L'application d'un

algorithme d'optimisation similaire à celui utilisé pour le choix des hyperparamètres du MLP (section 3.5.1) n'a pas permis d'améliorer significativement les performances de ce modèle.

Les résultats obtenus près de la côte sont similaires à ceux trouvés en utilisant un réseau MLP simple avec une ACP en entrée. Les erreurs au niveau des inondations sont légèrement plus basses sur les scénarios extrêmes. Une comparaison plus approfondie des erreurs obtenues avec les différents modèles sera présentée plus loin.

5.2 Réseau à entrée de type Vnet

Le second type de réseau encodeur essayé est un réseau de type Vnet [20]. Celui-ci transforme la grille d'entrée sans changer sa taille. On peut le décomposer en deux parties : la première partie est semblable au réseau présenté en section précédente, on effectue des blocs de convolutions suivis de réductions de dimension tout en augmentant le nombre de canaux. Une première différence est qu'ici l'opération de *max-pooling* est remplacée par une convolution avec un espacement de deux, ce qui a toujours pour effet de diviser les dimensions de l'image par deux en effectuant une compression moins brutale. La deuxième différence est l'ajout de raccourcis (*skip-connections*) entre l'entrée et la sortie de chaque bloc de convolutions. Ces raccourcis consistent simplement à additionner la sortie de chaque bloc avec son entrée, élément par élément, et permettent une meilleure propagation de l'information.

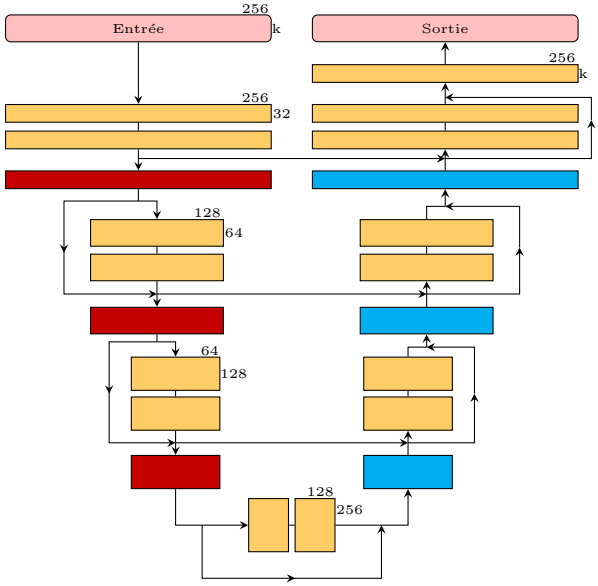


FIGURE 23 – Exemple de réseau Vnet à quatre niveaux. En rouge les convolutions avec un *stride* de deux, en bleu les convolutions transposées. La hauteur des rectangles représente le nombre de canaux (en particulier $k = 3$ pour une image RGB), leur largeur la taille de l'image.

La seconde partie du modèle est un miroir de la première : on effectue toujours des blocs de convolutions accompagnés de raccourcis, les opérations de réductions étant ici remplacées par des convolutions transposées effectuées avec un espacement de 2, qui ont pour effet de doubler la taille de l'image traitée. Dans ce cas, une convolution transposée revient à faire une convolution classique sur une version étendue de l'image traitée, des coefficients nuls étant rajoutés entre les coefficients de l'image pour doubler ses dimensions. On ajoute de plus des raccourcis reliant la partie encodeuse (descendante) et la partie décodeuse (montante) à chaque niveau de compression. La figure 23 offre une représentation graphique de cette architecture.

L'image obtenue à la sortie du Vnet a donc les mêmes dimensions que l'image d'entrée, on peut voir cette étape comme un prétraitement de la grille d'entrée. Ce type d'architecture est habituellement plutôt utilisé pour la segmentation d'images, par exemple dans le domaine médical [20]. Des architectures similaires sont également utilisées pour des problèmes de translation d'image à image [12], où le but est de transformer des images d'un certain type de sorte à ce qu'elles prennent les caractéristiques d'un autre type d'image. On peut

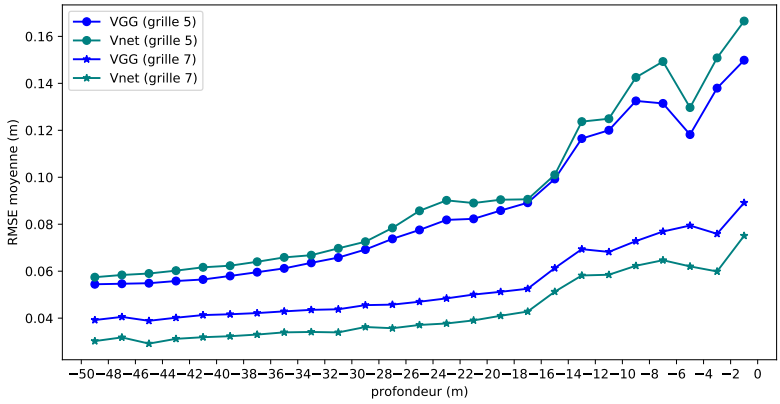


FIGURE 24 – Erreur selon la profondeur pour les scénarios à forte amplitude du jeu de validation de Nice

par exemple transformer une image satellite en un plan, ou vis-versa. Les images et leurs transformations gardent en générale la même structure spatiale sous-jacente, ce qui n'est vrai dans le cas des grilles mères et filles : une cellule de la grille mère correspond au minimum à 90^2 cellules de la grille fille.

Le modèle Vnet suivi de couches denses en sortie offre des performances très proches de celles du VGG. La figure 24 donne les erreurs obtenues sur les grilles 5 (aéroport) et 7 (Saint-Laurent-du-Var) du site de Nice pour les scénarios de forte amplitude. Les différences entre les erreurs obtenues sont faibles (généralement inférieures à $1cm$), et l'on note que le VGG surpasse le Vnet sur la grille 5 mais est légèrement moins efficace sur la grille 7. Le Vnet semblant présenter des performances similaires à celles du VGG pour une complexité bien supérieure, on privilégiera le VGG pour les prédictions de hauteurs d'eau en grille fine. Cependant, un réseau Vnet composé uniquement de couches de convolutions sera utilisé dans la section 7 pour prédire des grilles mères à l'aide de conditions initiales. Dans cette dernière application une correspondance spatiale directe existe entre les grilles d'entrées et de sortie (elles sont superposables), ce qui n'est pas le cas ici.

5.3 Influence du choix de la grille d'entrée

Deux grilles d'entrée de tailles variables ont précédemment été testées dans le cadre de l'étude de paramètres menée sur la base de données de Cannes. On considère ici le second jeu de données, qui utilise des grilles d'une résolution supérieure ($900m$ au lieu de $3700m$). Il devient alors nécessaire de redimensionner la grille d'entrée si l'on souhaite conserver la même étendue géographique ($(150 \times 3.7)^2 = 308025km^2$ pour la plus petite), au prix d'une interpolation. Cependant, l'étude semblait montrer qu'une plus grande étendue de la grille d'entrée n'apportait pas de gain de précision. On essaie ainsi plusieurs nouvelles grilles tronquées sans interpolation, de dimensions variables (voir le détail en section 2.4). Le tableau 4 compare les erreurs RMSE globales de validation sur la grille 5 (aéroport de Nice) obtenues sur une moyenne de trois entraînements, pour des modèles de type VGG et MLP (avec ACP en entrée). Les numéros des grilles mères considérées correspondent à ceux renseignés sur la figure 6. On observe peu de différences selon les entrées choisies. Les grilles très resserrées sur la côte offrent des performances semblables à des grilles nettement plus larges. Les comparaisons d'erreurs effectuées précédemment sur Cannes sur plusieurs modèles pour des grilles de tailles variables semblent également indiquer que ce comportement n'est pas uniquement dû au choix du modèle. Dans le cas d'un apprentissage direct associant grille mère et grille fille sur un site géographique fixé, l'information située au plus proche des côtes sur la grille mère semble donc suffisante pour obtenir des prédictions

Grille mère	Dimensions	MLP	VGG
G1	64×64	0.06507	0.06591
G2	128×128	0.06449	0.06651
G3	128×256	0.06619	0.0674
G4	256×256	0.06658	0.06611
G5	256×256 (résolution à $1800m$)	0.06463	0.06603

TABLE 4 – Erreurs RMSE globales (en m) entre $-50m$ et $+2m$ selon la grille d'entrée

Modèles	Grilles à $3700m$	Grilles à $900m$
Modèle 1	0.07459	0.07626
Modèle 3 (ACP)	0.06439	0.06637

TABLE 5 – Erreurs de *cross-validation* pour deux modèles MLP sur Cannes (1 couche, 1024 neurones)

convenables sur la grille fille.

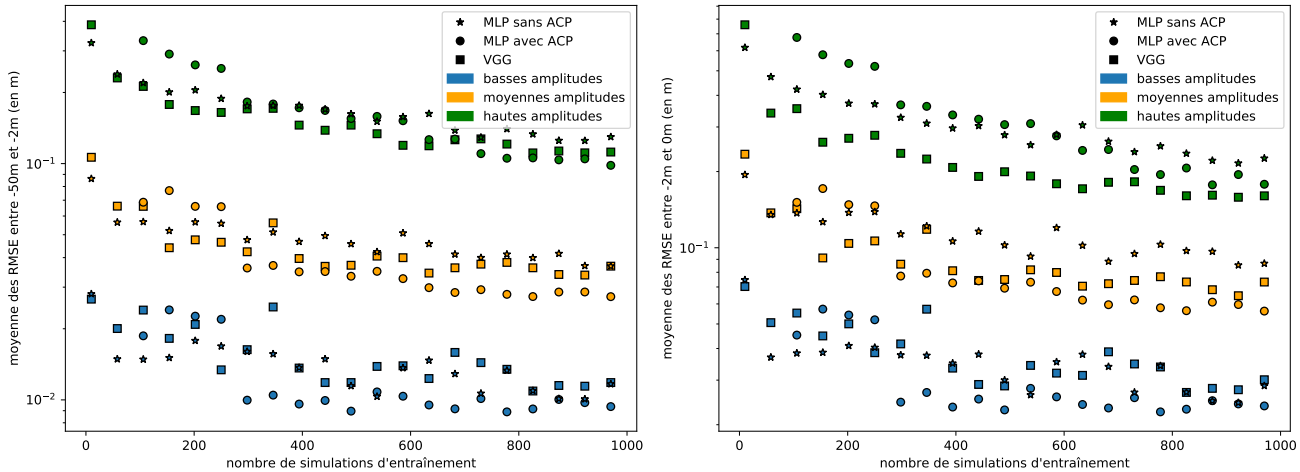
La résolution de la grille mère pourrait également avoir un impact sur la qualité de l'apprentissage. On essaie ainsi d'entraîner un réseau MLP sur la base d'entraînement de Cannes en utilisant des grilles mère à une résolution de $900m$. Une partie des grilles mères est ainsi regénérée à l'aide du code Taitoko à une résolution supérieure en utilisant les mêmes paramètres de source. Les erreurs relevées pour ce nouvel entraînement sont très semblables à celles obtenues précédemment (voir le tableau 5), et le changement de résolution ne semble pas apporter de différence notable. On note cependant que seules les grilles mères ont été recalculées par souci d'économie, les grilles filles de la base d'entraînement restent celles calculées à l'aide d'une grille mère de résolution $3700m$.

5.4 Impact de la base d'apprentissage

5.4.1 Évolution de l'erreur selon le nombre de scénarios

La durée d'entraînement des réseaux considérés reste raisonnable, pour autant que l'on dispose d'une accélération par GPU (voir la section 5.5. Cependant, la génération des jeux de données est particulièrement coûteuse : il faut effectuer des centaines de simulations, une simulation pouvant durer une dizaine de minutes sur 168 coeurs avec les paramètres choisis. On peut ainsi chercher à évaluer l'impact de la taille de la base d'apprentissage en faisant varier le nombre de scénarios d'entraînement et en traçant l'erreur correspondante sur le jeu de validation pour chaque entraînement. Il faut cependant déterminer une méthode d'échantillonnage pour sélectionner les nouveaux scénarios utilisés à chaque étape.

On se place sur la base de données de Nice qui offre davantage de scénarios, ce qui permettra d'observer l'évolution de l'erreur sur davantage de points. Choisir les nouveaux scénarios uniformément sur l'ensemble des simulations ne semble pas raisonnable. Il pourrait être intéressant de choisir les nouveaux scénarios de sorte à maximiser la variabilité des paramètres sources utilisés pour chaque simulation. Cela ne garantirait cependant pas une grande diversité de grilles filles. Le choix a donc été fait d'essayer de conserver la même répartition des amplitudes que dans la base de données complète. Les scénarios sont alors divisés en scénarios à basses, moyennes et hautes amplitudes selon la méthode détaillée précédemment en section 4.1.



(a) Entre -50 et $-2m$ de profondeur

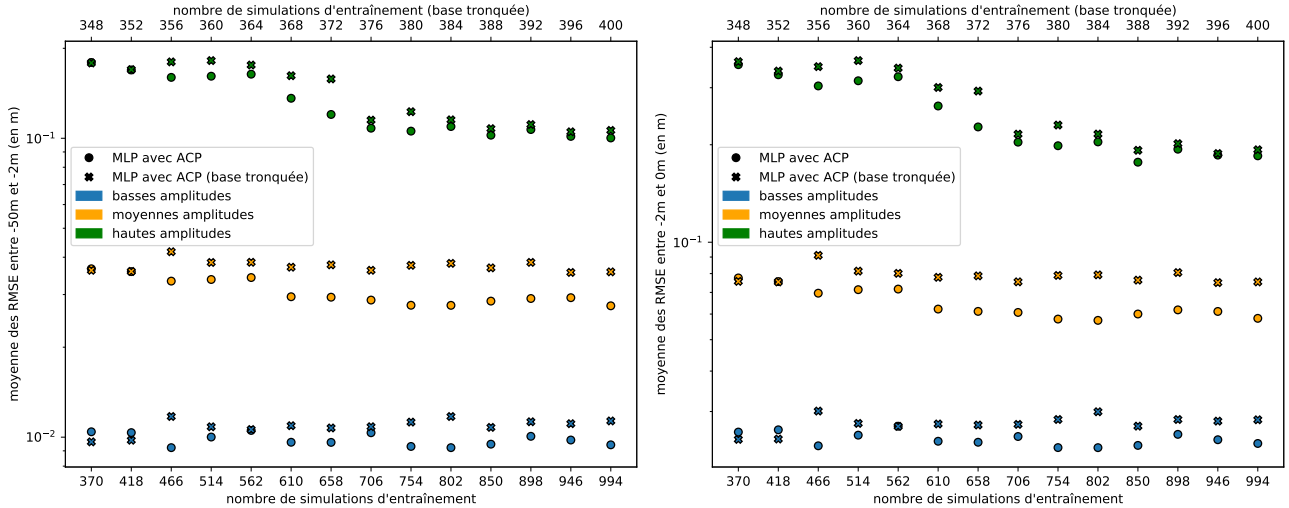
(b) Entre -2 et $0m$ de profondeur

FIGURE 25 – Erreur moyenne sur le jeu de validation de Nice en fonction du nombre de scénarios d'entraînement

La figure 25 présente l'évolution des erreurs sur le jeu de validation de Nice en profondeur en fonction du nombre de scénarios d'entraînement. On considère d'une part l'erreur au plus proche de la côte (entre $-2m$ et $0m$) et d'autre part l'erreur moyenne sur une bande de profondeur plus large (entre $-50m$ et $-2m$). On note que contrairement aux attentes le modèle de convolution présente une erreur plus basse que celle du MLP lorsque le nombre de simulations est réduit. De plus, l'utilisation d'une ACP ne permet pas d'obtenir de meilleurs résultats lorsque le nombre de scénarios est trop réduit. La tendance se renverse lorsque le nombre de scénarios dépasse les 300, ce qui se rapproche de la taille de la base de données de Cannes.

L'échantillonnage joue probablement un rôle important dans ces observations. On voit en effet une chute brutale de l'erreur vers les 300 scénarios, qui est sûrement due au choix des quelques scénarios ajoutés à cette étape. Les grilles de sortie sur Cannes et Nice présentent des caractéristiques différentes (taille, amplitude des simulations ...). La distribution des paramètres des scénarios sur le jeu d'entraînement limité de Cannes favorise potentiellement davantage l'utilisation d'une ACP que dans le cas du jeu de Nice, ce qui expliquerait l'écart limité des performances avec et sans ACP.

Enfin, on note une certaine stagnation des erreurs pour les scénarios à moyennes et basses amplitudes après avoir atteint les 300 scénarios, tandis que l'erreur pour les scénarios à forte amplitude continue de descendre. Ceci peut encore être dû à l'échantillonnage. Toutefois, cette dernière catégorie rassemblant les évènements rares à très hautes amplitudes, il est probable que les modèles aient besoin de davantage d'exemples d'apprentissage pour pouvoir les prédire, tandis que les autres catégories sont déjà suffisamment représentées. Pour confirmer cette observation, on choisit de réentraîner un MLP en incrémentant le nombre de scénarios de la même façon que précédemment, mais en cessant d'ajouter des scénarios à moyenne ou basse amplitudes après avoir constitué une base de plus de 346 scénarios. On continue d'ajouter les mêmes scénarios à hautes amplitudes que dans le cas précédent. La figure 26 compare les erreurs obtenues en utilisant la base d'entraînement tronquée par cette méthode, et la base de données complète. Les nombres de scénarios correspondant à chaque entraînement sont reportés sur les deux échelles en abscisse. Les tracés d'erreurs semblent vérifier les observations précédentes : les erreurs pour les scénarios à hautes amplitudes sont similaires aux erreurs obtenues précédemment, tandis que les erreurs trouvées pour les scénarios à moyennes et basses amplitudes ne sont que légèrement dégradées. On obtient ainsi des erreurs très similaires en utilisant 400 ou 994 scénarios d'entraînement. Il semble donc possible d'obtenir une base d'entraînement viable en utilisant un nombre très réduit de scénarios. De plus, obtenir des grilles filles précises est crucial dans le cadre des scénarios présentant de fortes amplitudes de vagues. On pourrait donc choisir de sacrifier quelques centimètres de précision dans les autres cas de figure pour augmenter la proportion de scénarios forts. Augmenter trop fortement cette proportion pourrait cependant mener à une importante surestimation des hauteurs d'eau sur les scénarios à moyennes et basses amplitudes.



(a) Entre -50 et $-2m$ de profondeur

(b) Entre -2 et $0m$ de profondeur

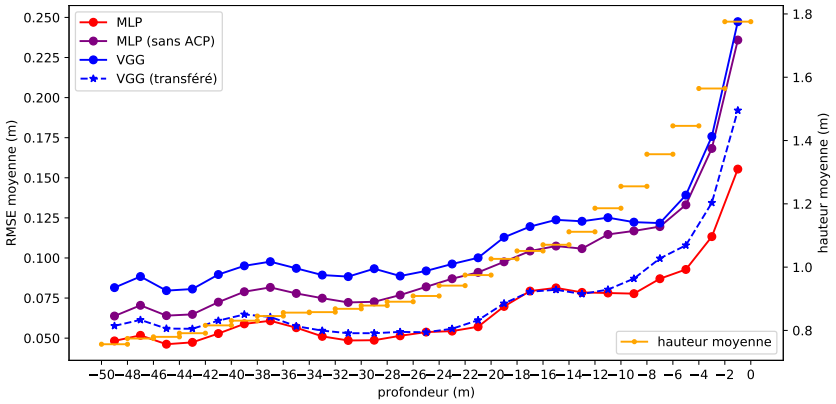
FIGURE 26 – Erreur moyenne sur le jeu de validation de Nice en fonction du nombre de scénarios d'entraînement

5.4.2 Transfert d'apprentissage

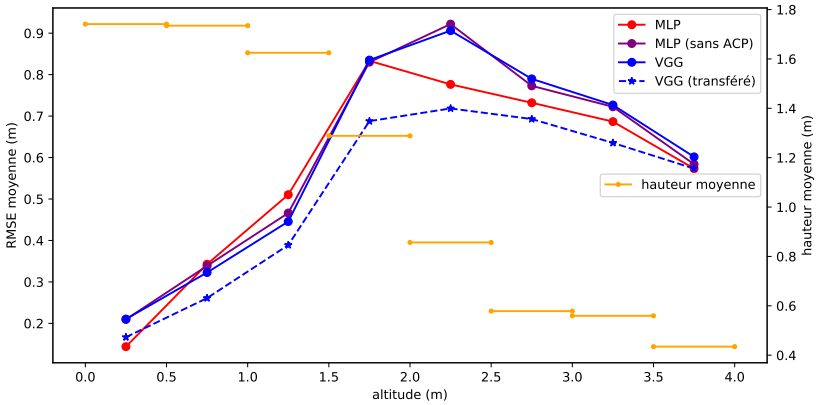
Le nombre de scénarios d'entraînement est sévèrement réduit pour la base de données centrée sur Cannes comparé à celui de Nice. Il peut donc être intéressant d'essayer de récupérer l'information apprise par un modèle sur la base de Nice et de la transmettre à un modèle entraîné sur la base moins importante de Cannes. Les données d'entrée appartiennent au même domaine dans les deux cas, pour autant que l'on utilise la même résolution de grille mère. Les tâches d'apprentissage sont différentes, les données de sortie appartenant à des domaines différents, mais les deux tâches d'apprentissage présentent des similarités.

Une approche simple au problème est d'entraîner sur le site de Cannes un modèle préentraîné sur Nice. On choisit pour cela un modèle de type VGG, entraîné sur la grille de sortie 5 de Nice, celle-ci pouvant présenter des amplitudes et des inondations importantes. Les couches de convolution situées à l'entrée du modèle ont appris les caractéristiques essentielles des grilles mères, on souhaite donc conserver leurs poids. On peut ensuite choisir différentes approches pour traiter les couches denses. On peut choisir de conserver les deux couches denses et rajouter de nouvelles couches à la suite de sorte à obtenir la bonne dimension de sortie. Il serait cependant nécessaire d'ajouter au minimum deux couches supplémentaires, les connexions entre les couches de sortie de Nice et Cannes ajoutant un nombre de paramètres déraisonnable (plus de sept milliards). Au vu de la perte d'efficacité observée des réseaux dans le cas de MLP trop profonds, on choisit plutôt de remplacer totalement la couche de sortie sans augmenter la profondeur du réseau. La couche cachée peut être au choix conservée et réentraînée ou supprimée. Une comparaison numérique semble indiquer qu'il n'y a pas de bénéfice à conserver les poids de cette couche cachée.

Les poids des couches de convolution sont fixés durant ce nouvel entraînement, qui ne porte que sur les poids des couches denses. On peut éventuellement les réentraîner sur quelques itérations pour avoir un ajustement plus précis (*fine tuning*) avec un risque de surapprentissage accru.



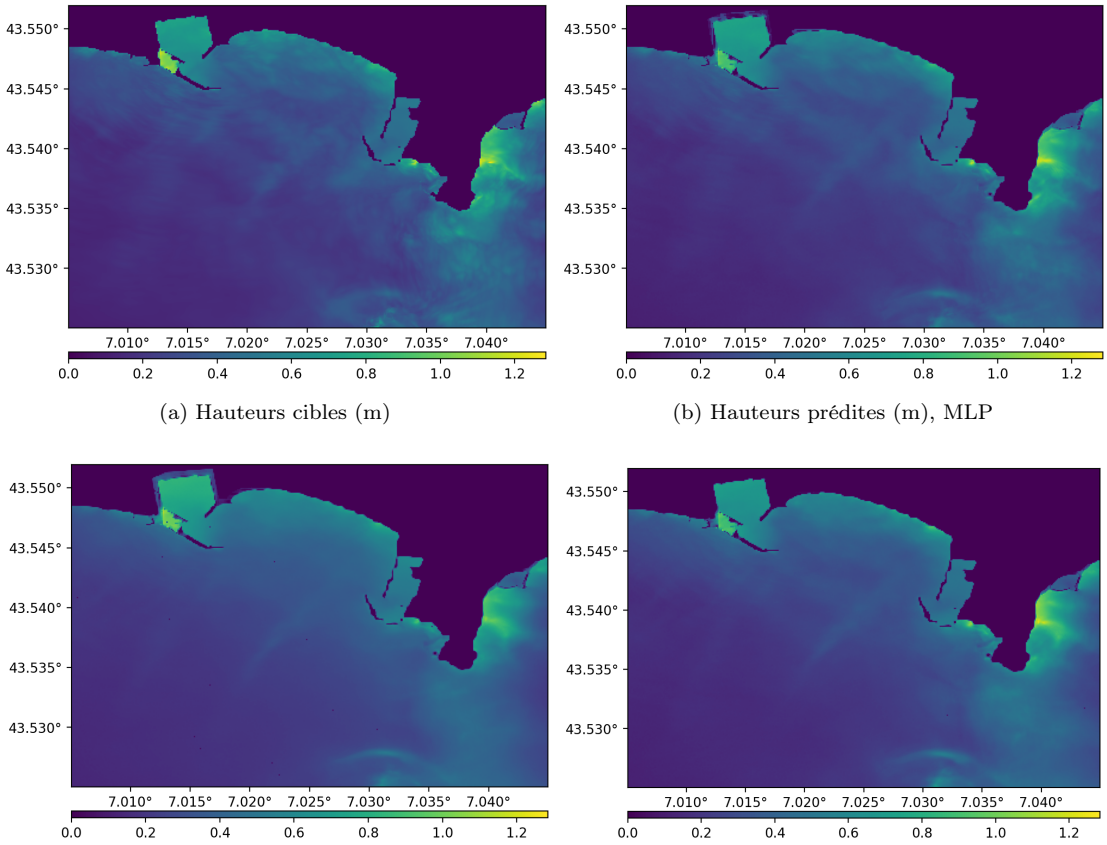
(a) Erreurs en profondeur



(b) Erreurs d'inondations

FIGURE 27 – Erreurs selon la profondeur sur Cannes (amplitudes fortes)

On observe des résultats intéressants pour une technique aussi basique : les résultats sont globalement meilleurs qu'en entraînant directement un VGG complet uniquement sur la base de Cannes. Les erreurs obtenues sont plus basses que celles trouvées pour un réseau MLP sans prétraitement, mais restent globalement légèrement supérieures au cas du MLP avec ACP en profondeur. Les résultats semblent cependant meilleurs au niveau des inondations sur les scénarios à fortes amplitudes. La figure 27 compare les erreurs des différents modèles selon la profondeur, pour les scénarios à fortes amplitudes du jeu validation. On note peu de différences avec le MLP en profondeur, l'écart se creusant aux faibles profondeurs. On observe le même type de comportement pour les scénarios à moyennes et faibles amplitudes. La figure 28 donne un exemple de prédiction favorable au VGG transféré.



(a) Hauteurs cibles (m)

(b) Hauteurs prédites (m), MLP

(c) Hauteurs prédites (m), VGG

(d) Hauteurs prédites (m), VGG avec transfert

Modèle	RMSE (-50/2m)	RMSE (-2/0m)	RMSE (0/2m)
VGG transféré	0.04405	0.0663	0.14583
VGG	0.06254	0.08091	0.20954
MLP sans ACP	0.09203	0.10981	0.37485
MLP	0.0445	0.0506	0.18234

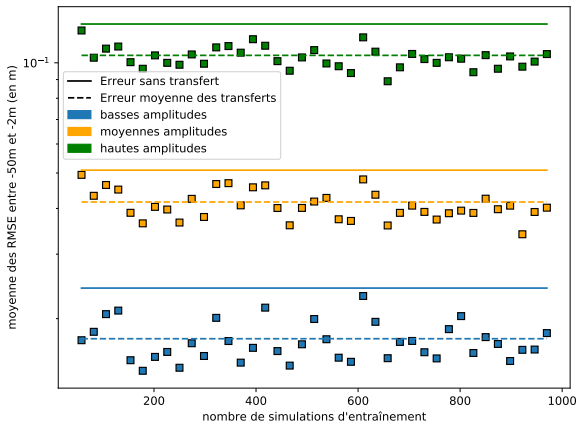
(e) Erreurs RMSE (m)

FIGURE 28 – Hauteurs de vague pour un séisme de magnitude 6.9 en zone 5

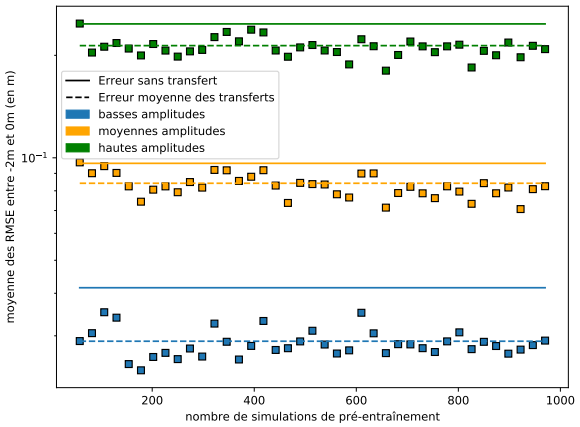
La section précédente évaluait l'influence du nombre de scénarios sur la qualité de l'apprentissage d'un modèle. On peut effectuer la même étude ici, en faisant cette fois varier le nombre de scénarios d'apprentissage du modèle avant son transfert. Le modèle est donc préentraîné sur un nombre réduit de scénarios, puis est réentraîné sur la base de données complète de Cannes, les poids des couches de convolution étant comme avant fixés. Dans le cas d'un transfert efficace, l'erreur devrait diminuer avec le nombre de scénarios, comme observé en partie précédente. Les résultats observés sont cependant assez différents.

La figure 29 présente les erreurs obtenues sur les jeux de validation de Cannes (grille 1) et d'Antibes (grille 6, plage de Juan les Pins) pour différents nombres de scénarios de pré-entraînement, c'est-à-dire de

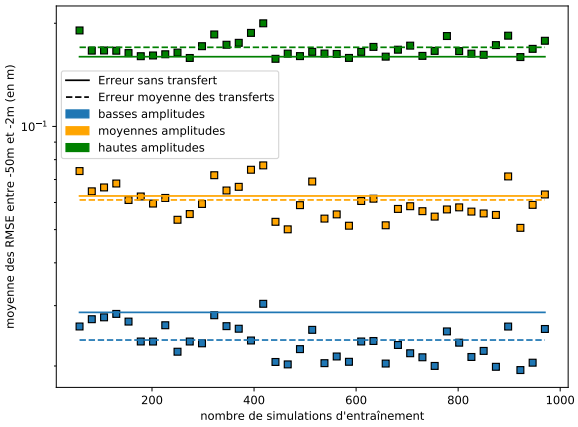
scénarios utilisés pour entraîner le VGG sur Nice avant son transfert. On observe peu de progrès lorsque le nombre de scénarios augmente, et des oscillations de l'erreur selon les nouveaux scénarios ajoutés. Les lignes en traits plein donnent les erreurs obtenues par un VGG entraîné directement sans transfert, et les lignes en pointillés donnent la moyenne des erreurs trouvées pour les modèles transférés. En moyenne, les modèles transférés semblent offrir de meilleures performances que le modèle entraîné directement sur le jeu de Cannes, mais l'augmentation du nombre de simulations de pré-entraînement ne permet pas de creuser l'écart. Le jeu d'Antibes, qui est constitué des mêmes scénarios que celui de Cannes, ne présente pas les mêmes résultats. Aucune amélioration des performances n'est observable dans ce cas, la moyenne des erreurs des modèles transférés dépassant même légèrement l'erreur obtenue sans transfert. Ce comportement peut être dû à un meilleur apprentissage du modèle complet sur cette zone, qui dépasse le léger gain apporté par le transfert. Cette méthode de transfert est dans tous les cas peu efficace et les gains réels sont peu probants.



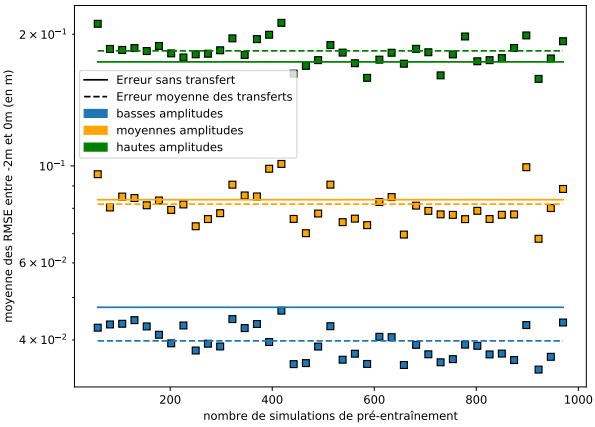
(a) Entre -50 et $-2m$ de profondeur (Cannes)



(b) Entre -2 et $0m$ de profondeur (Cannes)



(c) Entre -50 et $-2m$ de profondeur (Antibes)



(d) Entre -2 et $0m$ de profondeur (Antibes)

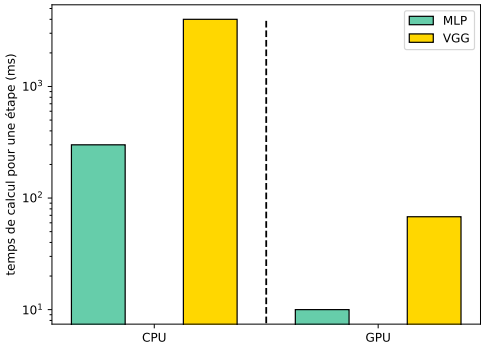
FIGURE 29 – Erreur moyenne sur les jeux de validation de Cannes (grille 1, voir la figure 6) et Antibes (grille 6, figure 51) en fonction du nombre de scénarios d'entraînement

5.5 Temps de calcul

Les études de la section 3 ont été effectuées sur un ordinateur possédant 56 coeurs de calculs, en utilisant uniquement la parallélisation sur CPU (*Central Processing Unit*). Une étape de calcul (traitement d'un *batch* de 32 grilles d'entraînement) sur le jeu d'entraînement de Cannes prenait environ $350ms$ pour un MLP sans ACP, soit un peu plus de 15 minutes pour un entraînement complet de 300 itérations, chaque itération comportant dans ce cas dix étapes. Un passage sur la base de données de Nice, plus conséquent, aurait augmenté le temps de calcul à près d'une heure par entraînement en utilisant les mêmes paramètres. Le lancement d'une recherche d'hyperparamètres à 50 entraînements aurait ainsi pris près de deux jours de

calcul. Le temps de calcul augmente également considérablement avec l'ajout de couches de convolution : une étape d'entraînement d'un VGG prend plus de quatre secondes sur le même ordinateur, soit trois heures pour un entraînement.

La mise à disposition d'un accès au supercalculateur COBALT situé au sein du TGCC (Très Grand Centre de Calcul) a permis de grandement accélérer la suite de l'étude, et surtout de l'étendre aux architectures à couches de convolution. Les noeuds de calculs utilisés au cours du stage sont spécialisés pour les applications de *deep learning* et comportent chacun quatre cartes graphiques Nvidia Tesla V100-SXM2 ainsi que 40 coeurs de calcul. L'accélération sur carte graphique (GPU ou *Graphics Processing Unit*) incorporée au sein de la librairie Tensorflow permet de décupler la vitesse d'entraînement, particulièrement pour les modèles à couches de convolution. La figure 30 compare les temps de calcul moyens pour une étape d'entraînement (traitement d'un *batch*) et pour un entraînement (300 itérations comportant 36 étapes chacune) sur Nice. Les grilles d'entrée et de sortie choisies sont de taille 128×256 (grille d'entrée 3 et grille de sortie 1). L'entraînement sur 300 itérations d'un VGG dure ainsi environ 11 minutes en utilisant une accélération par GPU, tandis qu'un entraînement équivalent sur CPU prend plus de dix heures. Le même constat est vérifiable pour le modèle MLP, même si le facteur d'accélération est moins important. Le temps de prédiction reste faible, même sur CPU. La prédiction d'un scénario dure environ $100ms$ sur CPU pour le VGG, et moitié moins pour un modèle MLP.



(a)

	CPU		GPU	
Temps	MLP	VGG	MLP	VGG
Une étape (ms)	300	4000	10	70
300 itérations (min)	48	640	1.6	11.2

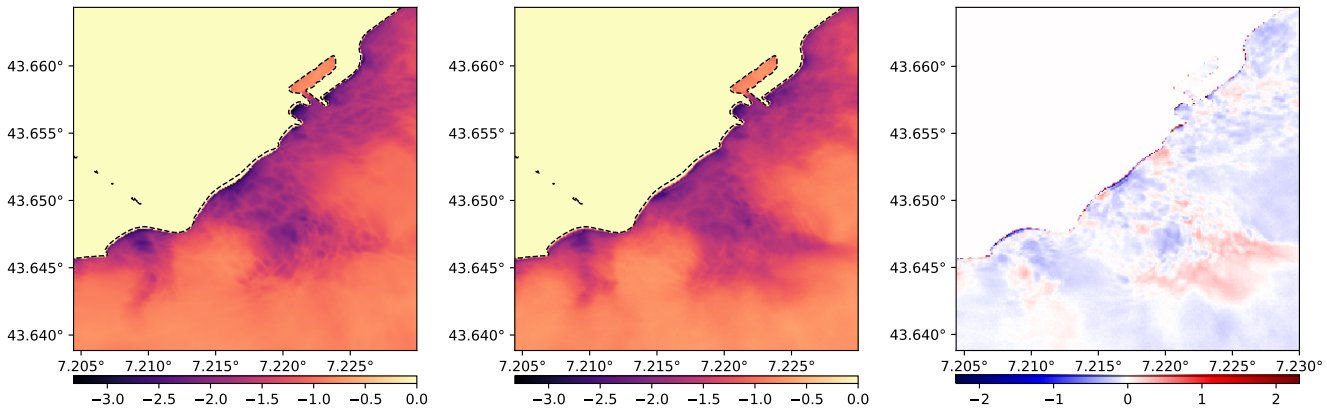
(b)

FIGURE 30 – Comparaison des temps de calcul avec et sans utilisation d'une accélération sur GPU pour les grilles d'entrée 3 et de sortie 1 sur Nice (voir la figure 6)

6 Prédiction des inondations et des retraits

6.1 Prédiction des hauteurs minimales

Dans le cadre du jeu de donnée de Nice, les grilles mères et filles des hauteurs minimales de vagues ont également été calculées. On peut donc s'intéresser à l'information apportée en entrée par les grilles mères de hauteurs minimales, en particulier dans le cadre de réseaux de convolutions capables de "superposer" plusieurs grilles. D'un autre côté, on peut aussi s'intéresser à la capacité des modèles à prédire ces hauteurs minimales sur la grille fille, ainsi que le retrait des vagues qui y est associé.



(a) Hauteurs cibles (m) (b) Hauteurs prédites (m), VGG (c) Erreur signée (m)

FIGURE 31 – Prédiction des hauteurs minimales sur la grille 5 (figure 6) de Nice pour un séisme de magnitude 7.2 en zone 5

La figure 31 donne un exemple de prédiction des hauteurs minimales sur le jeu de validation, en utilisant la grille mère des hauteurs minimales en entrée. La RMSE entre $-2m$ et $0m$ de profondeur est de $22cm$ dans ce cas. Le trait de côte est tracé en pointillés. On observe un retrait de 10 à $20m$ en certains points de la côte.

On utilise les mêmes architectures qu'auparavant. On choisit de prédire les opposées des grilles de hauteurs minimales, ce qui permet de conserver une fonction d'activation RELU en couche de sortie. La grille tracée en figure 31 est donc l'opposée de la grille réellement prédite. Dans le cas d'utilisation simultanée des grilles de hauteurs maximales et minimales en entrée, on doit procéder différemment selon le type de modèle. Dans le cas du MLP, on aplatit simplement les grilles et on les concatène en un seul vecteur. Dans le cas des réseaux de convolution, on empile les deux grilles hmin et hmax pour obtenir une image à deux canaux. Dans le cas d'une grille de dimensions 256×256, on obtient ainsi un tenseur $256 \times 256 \times 2$. Cette dernière approche permet de combiner efficacement l'information contenue par les deux grilles : à chaque cellule de la grille mère sont associées une hauteur maximale et une hauteur minimale.

Le tableau 6 compare les erreurs obtenues en prédisant les hauteurs minimales sur la grille 5 de Nice à partir de différentes entrées et pour la prédiction des hauteurs maximales et minimales en utilisant un VGG. On note la présence d'artefacts (cellules nulles) en profondeur lors de la prédiction des hauteurs maximales

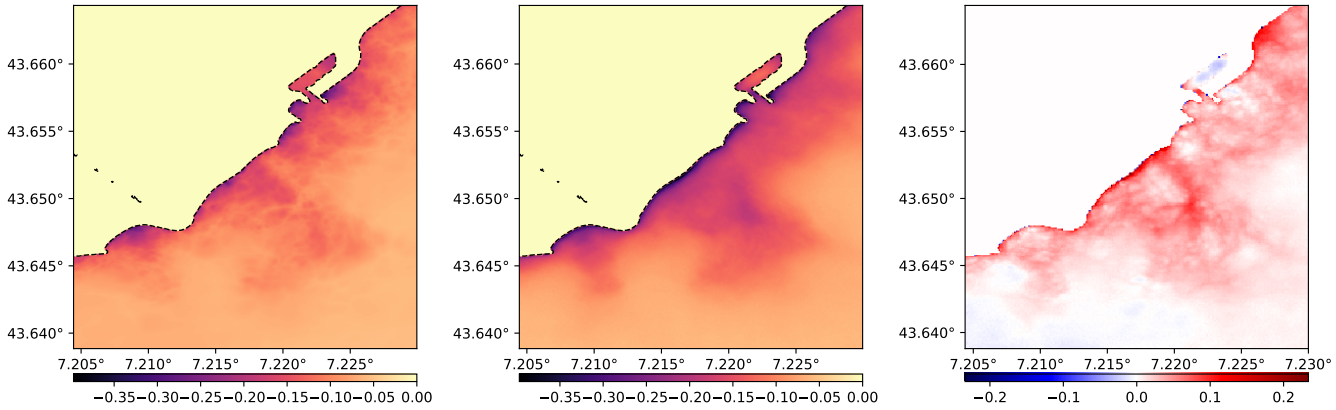
Entrées	RMSE (-2/0m)	R^2 (-2/0m)
hmin	0.11669	0.96247
hmin et hmax	0.11808	0.95866
hmax	0.11693	0.9655

(a) Prédiction des hauteurs minimales

Entrées	RMSE (-2/0m)	R^2 (-2/0m)
hmin	0.08254	0.96664
hmin et hmax	0.07362	0.97341
hmax	0.06964	0.97628

(b) Prédiction des hauteurs maximales

TABLE 6 – Erreurs RMSE globales (en m) et scores R^2 calculés entre $-2m$ et $0m$ de profondeur sur la grille 5 pour différents types d'entrées



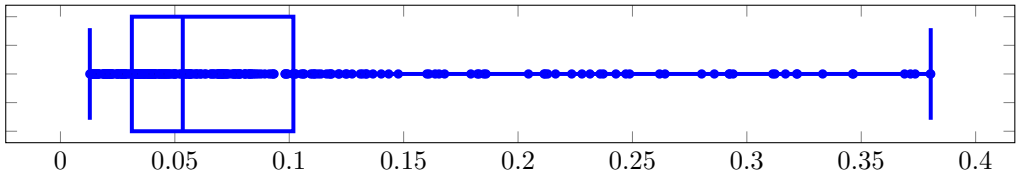
(a) Hauteurs cibles (m) (b) Hauteurs prédites (m), VGG (c) Hauteurs prédites (m), VGG

FIGURE 32 – Prédiction des hauteurs minimales sur la grille 5 (figure 6) de Nice pour un séisme de magnitude 7.7 en zone 3

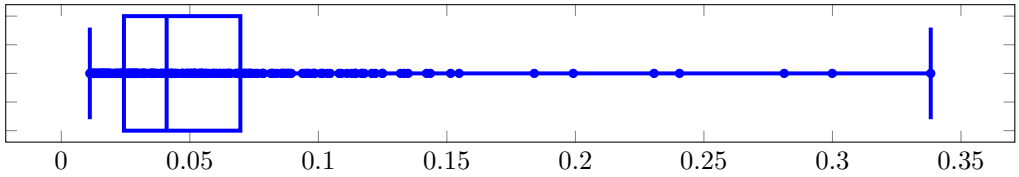
en utilisant les hauteurs minimales, ce qui cause une légère augmentation de l'erreur. En dehors de ce point, le modèle offre globalement des performances similaires quelque soit le type d'entrée, utiliser les deux grilles hmin et hmax simultanément en entrée ne semble pas améliorer significativement les performances du réseau.

Les prédictions des hauteurs minimales sont généralement moins précises que celles des hauteurs maximales pour les cellules situées sous le niveau de la mer, particulièrement pour celles situées au plus près de la côte. Les erreurs les plus importantes ne se limitent plus aux scénarios à hautes amplitudes de la zone 5. La figure 32 donne un exemple de prédiction de mauvaise qualité pour un scénario de zone 3. On relève dans ce cas une RMSE de $10cm$ entre $-2m$ et $0m$, pour une amplitude moyenne de $19cm$ à ces mêmes profondeurs dans la grille exacte.

La figure 33 donne la répartition des erreurs RMSE entre $-2m$ et $0m$ sur l'ensemble du jeu de validation de Nice pour la grille 5. On note que la distribution des erreurs des hauteurs minimales est plus étalée que celle des erreurs des hauteurs maximales au-delà du troisième quartile. On observe une variance plus importante ($8.4cm$ d'écart type contre $4.6cm$). Ces différences à faible profondeur s'expliquent par le fait que les points asséchés par le retrait des vagues présentent une hauteur de vagues nulle dans les simulations, ce qui rend les prédictions difficiles pour les réseaux et cause une augmentation importante de l'erreur. En effet, le calcul d'erreur est très punitif lorsqu'une cellule est prédite comme étant encore immergée mais avec une faible amplitude de vague. On retrouve le même type de problématique que dans le cas des inondations, avec la nuance que la bordure de la zone sèche est ici plus régulière et plus prévisible. Les prédictions ne présentent

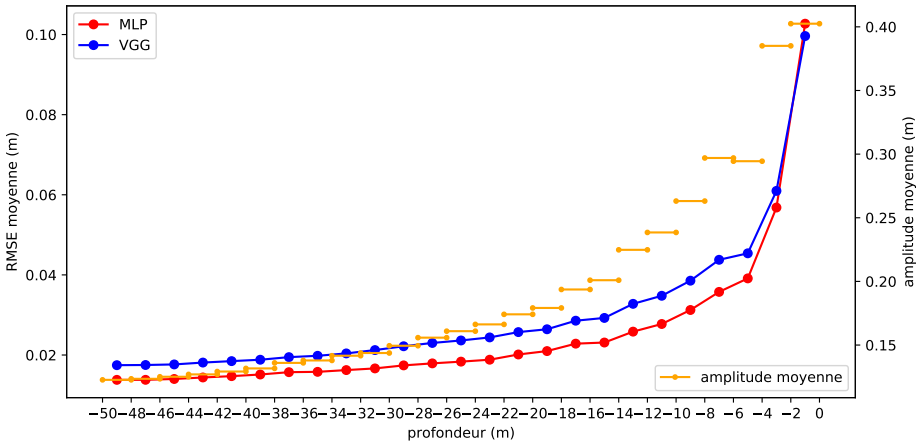


(a) Erreurs des hauteurs minimales

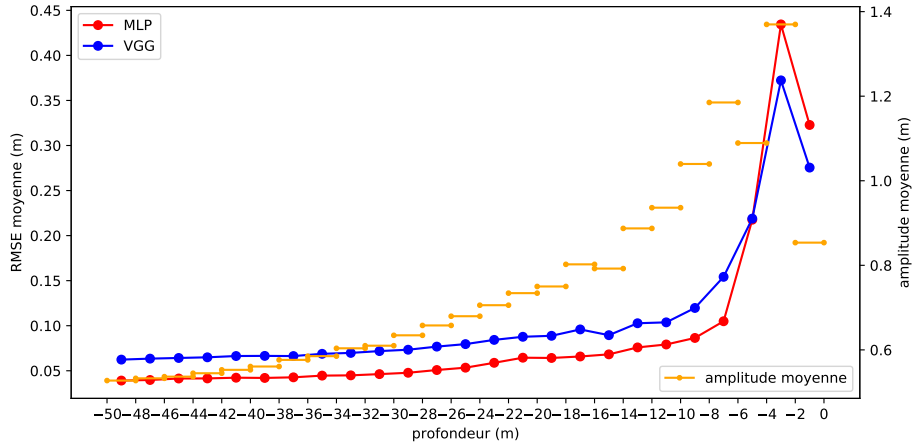


(b) Erreurs des hauteurs maximales

FIGURE 33 – Erreurs RMSE entre $-2m$ et $0m$ (en m) pour un VGG sur la grille 5 de Nice



(a) Moyennes amplitudes



(b) Hautes amplitudes

FIGURE 34 – Erreur selon la profondeur pour les hauteurs minimales (grille 5 de Nice)

ainsi pas de contours d'inondations très bruités, au contraire des prédictions de hauteurs maximales pour des scénarios à forte amplitude. La figure 34 présente le tracé de l'erreur selon la profondeur pour des modèles VGG et MLP. On note une augmentation brutale de l'erreur très proche des côtes, causé par le retrait des vagues pour les scénarios à fortes amplitudes. On note que dans ce cas l'amplitude moyenne proche de la côte est peu représentative. En effet, les points asséchés par le retrait des vagues présentent une hauteur minimale nulle, ce qui entraîne une diminution de l'amplitude moyenne à faible profondeur pour les scénarios à forte amplitude.

6.2 Prédictions des inondations

6.2.1 Ajout de scénarios amplifiés

La grille de sortie 5 du site de Nice, située au niveau de l'aéroport, présente sur certains scénarios extrêmes des inondations assez importantes. On peut parfois observer quelques centaines de mètres carrés d'inondation proches des côtes dans le cas de séismes à forte magnitude issus de la zone 5. Cependant les modèles peinent à reproduire ces inondations, notamment à cause d'un manque d'exemples d'entraînement, ces évènements restant des occurrences rares. Certaines zones occasionnellement inondées dans des scénarios de validation ne sont jamais inondées dans les prédictions du modèle. Pour fournir davantage d'exemples d'inondations fortes au modèle, on choisit de créer des scénarios artificiels à très forte magnitude. Ces scénarios sont improbables (voire impossibles) d'un point de vue géophysique, mais devraient permettre au modèle d'apprendre les motifs d'inondations.

Pour créer ces nouvelles simulations, on reprend des paramètres de source déjà présents dans le jeu d'entraînement, en modifiant uniquement la valeur du glissement dans le but d'augmenter la magnitude du séisme. On récupère ainsi les paramètres de source d'environ 10% des simulations d'entraînement issues de chaque zone. On augmente leur magnitude en gardant une borne supérieure de 7.9 pour les zones 5 et 6, et 8.5 pour les zones 3 et 4. On calcule ensuite le nouveau moment sismique à l'aide d'une loi empirique :

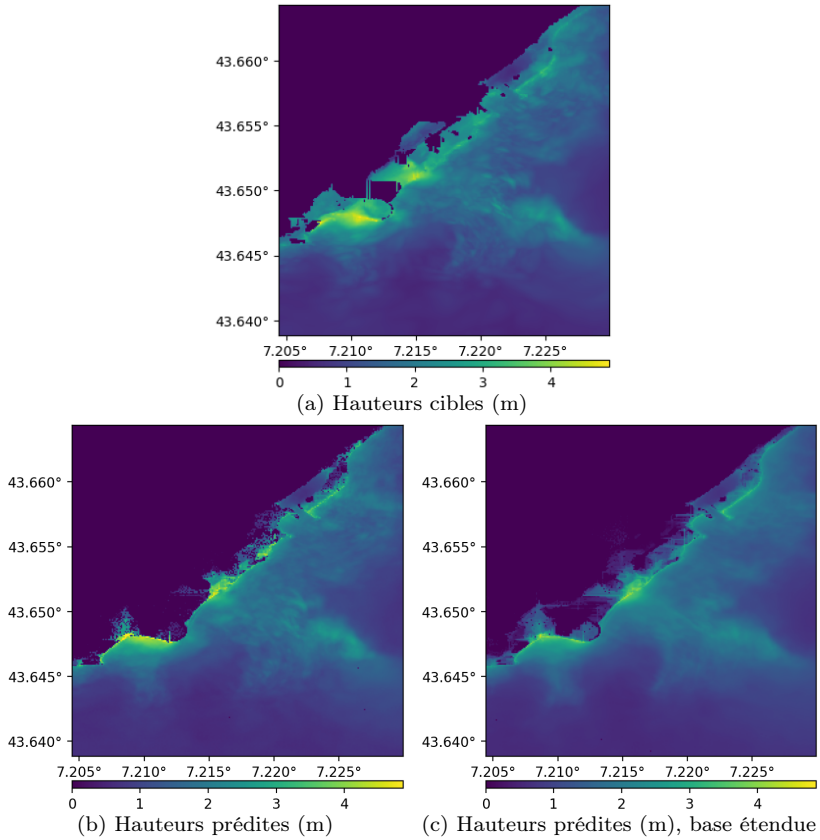
$$M_0 = 10^{\frac{3}{2}m+16.1} \tag{22}$$

où M_0 désigne le moment, m la magnitude. La nouvelle longueur de glissement peut enfin être calculée par la relation :

$$s = \frac{M_0}{A.R} \tag{23}$$

où s désigne la longueur de glissement (*slip*), A la surface de glissement et R la rigidité, ces deux derniers paramètres étant fixés dans les simulations de la base de données. On relance donc des simulations à l'aide du code Taitoko en se basant sur ces nouveaux paramètres. Une petite partie des simulations récupérées est mise à l'écart après inspection, les hauteurs de vagues calculées étant vraiment excessives. On garde finalement 85 scénarios à forte amplitude dont une bonne fraction présente des inondations importantes, pour certaines supérieures à toutes celles observées sur les jeux de validation et d'entraînement. La figure 35 présente un exemple de prédictions pour un modèle de type VGG, avec et sans ajout des scénarios amplifiés à la base d'entraînement.

Le modèle a tendance à prédire des inondations plus étendues, dépassant souvent les limites des inondations simulées, avec des hauteurs d'eau supérieures. Les inondations ont donc tendance à être soit sous-estimées dans les cas extrêmes, soit surestimées, et leur localisation laisse en général toujours à désirer. Cette capacité à inonder davantage réduit toutefois l'erreur sur une poignée de scénarios extrêmes. Dans d'autres exemples à amplitudes moindres, des zones originellement sèches sont inondées à tort par le modèle amplifié.

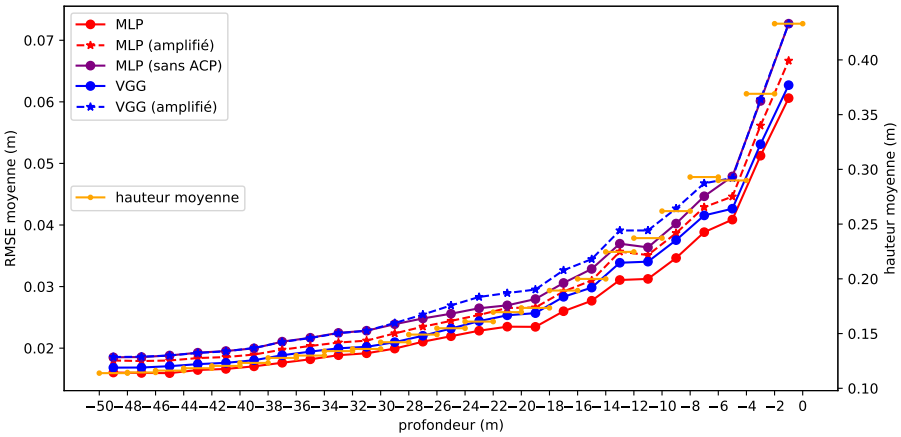


(a) Hauteurs cibles (m)

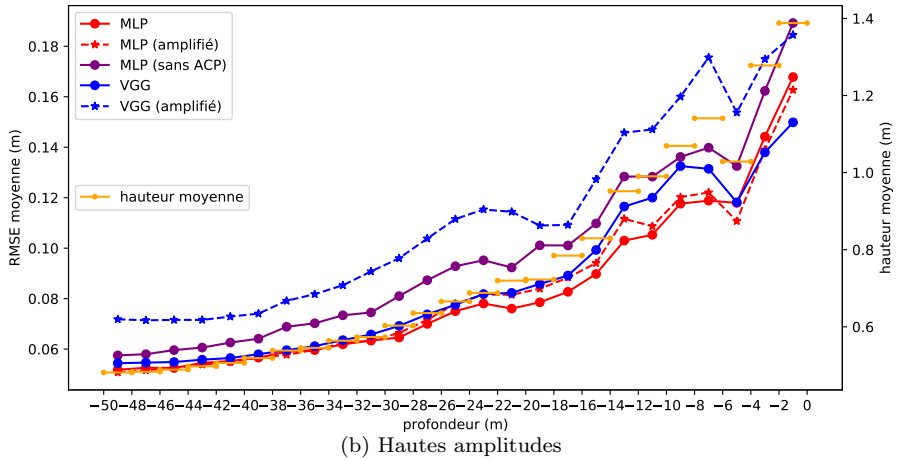
(b) Hauteurs prédites (m)

(c) Hauteurs prédites (m), base étendue

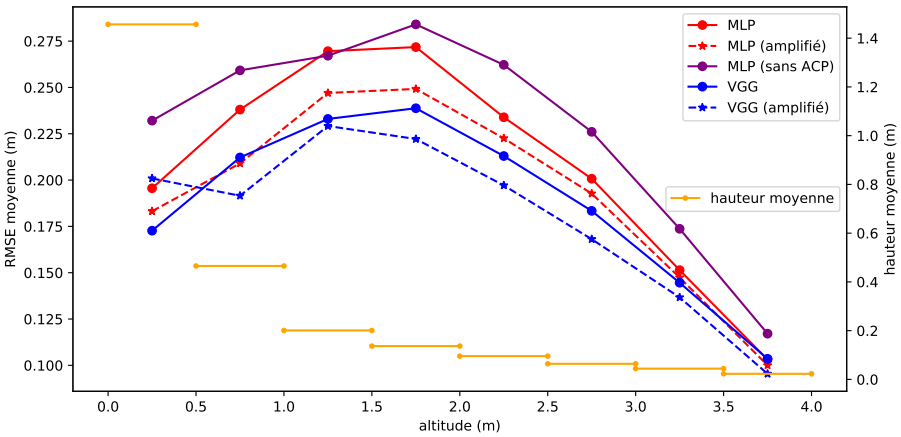
FIGURE 35 – Hauteurs maximales (m) simulées et prédites par VGG sur la grille fille 5 de Nice (figure 6) pour un magnitude 7.2 en zone 5



(a) Moyennes amplitudes



(b) Hautes amplitudes



(c) Hautes amplitudes (inondations)

FIGURE 36 – Comparaison des erreurs moyennes selon la profondeur

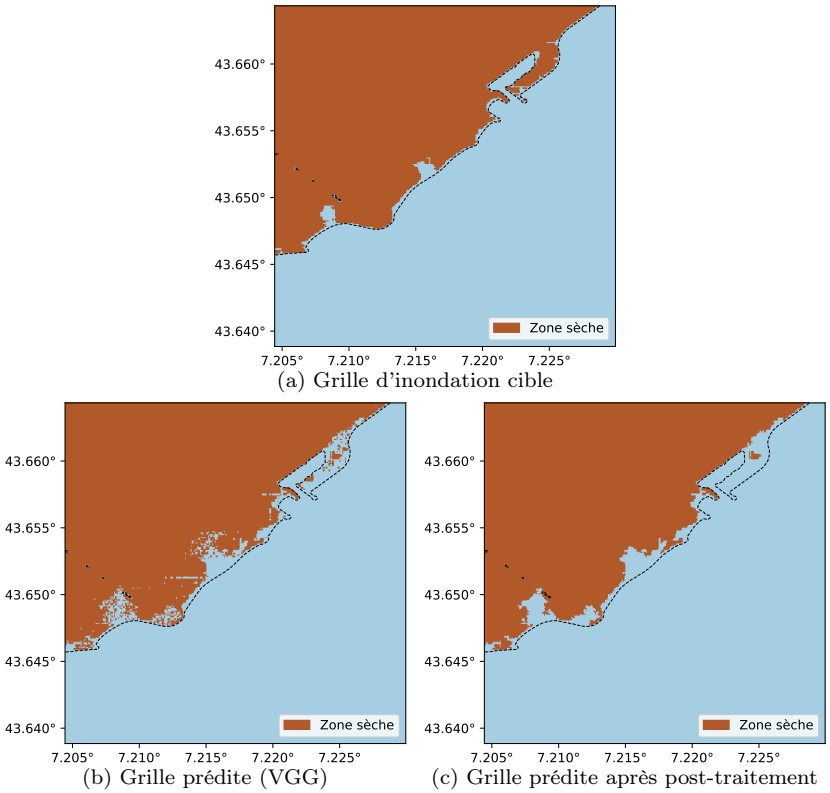
La figure 36 compare les erreurs en profondeur pour les différents modèles, avec ou sans la présence de scénarios amplifiés. Les prédictions en profondeur semblent dégradées après l'ajout de ces nouvelles simulations. Le VGG semble plus affecté que le MLP avec ACP en entrée. Le MLP affiche ainsi des erreurs réduites de quelques centimètres au niveau des inondations sur les scénarios extrêmes, avec une légère détérioration en profondeur. Le VGG devient au contraire moins performant qu'un MLP sans prétraitement en profondeur, mais est plus efficace pour les inondations.

Ajouter davantage de scénarios en sélectionnant spécifiquement des scénarios à inondations (certains scénarios amplifiés n'en présentent pas ou peu) pourrait améliorer la qualité de prédiction des inondations, au coût d'une surprédiction des hauteurs de vagues sur les scénarios non extrémaux. La sélection des paramètres de source de ces scénarios joue certainement un rôle important dans la qualité de l'apprentissage en résultant. Il est probable que la dégradation observée des performances en profondeur soit dûe à l'utilisation de paramètres trop éloignés de l'espace des configurations possibles en géophysique. Le problème est alors de sélectionner de nouveaux scénarios d'entraînement géophysiquement crédibles et présentant des inondations importantes.

6.2.2 Traitement des grilles d'inondations

S'il est intéressant de connaître la hauteur de vague au niveau des cellules inondées, on a vu que ces prédictions sont souvent de moins bonne qualité. On peut ainsi se limiter à observer uniquement le contour des zones inondées, et en particulier de l'inondation principale, une cellule étant considérée comme inondée lorsque la hauteur de vague en ce point est supérieure à vingt centimètres. Les grilles d'inondations résultantes donnent une estimation visuelle rapide de l'ampleur des zones touchées et peuvent présenter un intérêt en contexte d'alerte.

On note cependant que contrairement aux grilles de hauteurs minimales qui présentent toujours une ligne de retrait nette, les prédictions d'inondations sont beaucoup plus bruitées sur certaines grilles. Il est difficile de tracer un contour d'inondation clair, et il est nécessaire d'effectuer un post-traitement. On relève notamment



(a) Grille d'inondation cible

(b) Grille prédite (VGG)

(c) Grille prédite après post-traitement

FIGURE 37 – Grilles d'inondations sur la grille fille 5 de Nice (figure 6) pour un scénario de magnitude 6.8 en zone 5

la présence de nombreuses cellules non inondées au sein de zones inondées, et à l'inverse la présence de cellules inondées isolées au milieu de zones sèches.

Un algorithme de traitement assez basique a été mis en place pour obtenir des zones inondées et sèches plus régulières. On extrait les zones sèches (ensembles de plusieurs cellules sèches connectées) et l'on inonde celles qui présentent une aire sous un certain seuil, ce qui a pour effet de supprimer les groupements isolés de cellules sèches. On effectue ensuite la même opération sur les cellules inondées. On effectue enfin un dernier passage sur les cellules sèches. On récupère alors facilement l'ensemble des cellules sèches de la côte en prenant la plus grande zone sèche restante, et l'on en déduit la bordure d'inondation.

Concrètement, la récupération des zones sèches et inondées est implémentée à l'aide de l'algorithme *Union-Find*, qui permet d'extraire les composantes connexes sèches et inondées de la grille. La grille est parcourue cellule par cellule de haut en bas et de gauche à droite, les composantes connexes étant progressivement constituées au cours du parcours. Dans le cadre de l'algorithme *Union-Find* les composantes connexes sont représentées par des arbres. Par exemple, dans le cadre de l'extraction des zones sèches on attribue à chaque cellule sèche une cellule sèche parente dans l'arbre de la composante connexe correspondante. Chaque composante connexe est identifiée à fin de l'exécution de l'algorithme par un représentant unique correspondant à la racine de l'arbre. La figure 37 donne un exemple de grille d'inondation traitée par cette méthode.

Cet algorithme de post-traitement permet d'obtenir des grilles d'inondations plus nettes et plus lisibles dans les cas observés, mais présente des limites importantes. La segmentation par composantes connexes utilisée est arbitraire. Ainsi une zone présentant un nombre important de cellules inondées peut être ignorée si une bande continue d'une seule cellule de largeur la sépare de la mer. Au contraire la bordure de l'inondation principale peut aller profondément dans les terres pour peu qu'une bande de cellules inondée isole une masse de terre. Une autre façon de procéder serait de regarder localement au voisinage de chaque cellule le nombre de cellules sèches et inondées et d'utiliser une règle de décision plus poussée pour inonder ou assécher les cellules. On peut également rajouter une étape de lissage pour rendre le contour des inondations plus régulier et éliminer certains artefacts.

7 Modèle de prédiction complet

La méthode de prédiction des grilles filles présentée précédemment permet d'obtenir une grille haute résolution en moins d'une seconde. Cependant, ce temps de prédiction ne prend pas en compte le temps de calcul de la grille mère, qui prend un peu moins d'une minute sur un super-calculateur en employant plus d'une centaine de coeurs. Dans le cadre de simulations de grilles mères dans de grands bassins (par exemple l'Atlantique Nord-Est ou le Pacifique) le temps de calcul est allongé, la zone de calcul étant considérablement élargie. L'utilisation d'un réseau de neurones réduit ce temps de calcul à moins d'une seconde. Un deuxième avantage de cette méthode est la puissance de calcul employée : un ordinateur de bureau classique suffit à effectuer les prédictions, ce qui peut présenter un intérêt en cas de panne du calculateur.

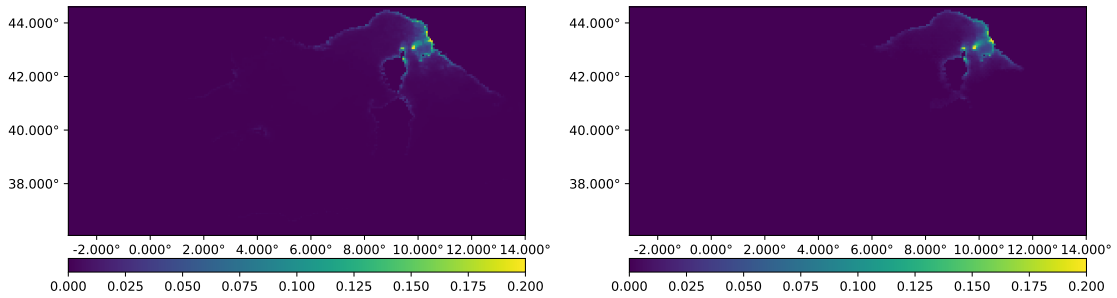
7.1 Prédiction des grilles mères

L'approche choisie est d'utiliser les conditions initiales de la simulation non linéaire pour prédire la grille mère complète à l'aide d'un réseau de convolution. On peut ensuite utiliser la grille ainsi générée pour effectuer une prédiction en haute résolution, pour un temps de calcul total inférieur à une seconde.

On procède de la manière suivante pour traiter les données. On récupère les grilles donnant les hauteurs de vagues des conditions initiales des simulations Taitoko. Ces grilles serviront d'entrées au réseau, tandis que les sorties seront les grilles mères obtenues après la propagation du Tsunami. Ces grilles sont tronquées de sorte à contenir uniquement les zones contenant les failles exploitées. On trouve ainsi des grilles 1024×2048 dont on divise la résolution par huit. On obtient finalement des grilles mères de dimensions 128×256, fortement sous-résolues ($7200m$ au lieu des $900m$ initiaux). Les dimensions sont alors des puissances de deux, ce qui permettra de bien définir l'architecture du réseau.

Le modèle de génération des grilles utilisé est un Vnet profond dont la description est donnée en section 5.2. Contrairement au cas d'utilisation précédent, on utilise uniquement des couches de convolution ici. Le réseau transforme une matrice 128×256 en une autre matrice de mêmes dimensions. On retrouve un cas d'application proche des problèmes de translation d'image à image [12].

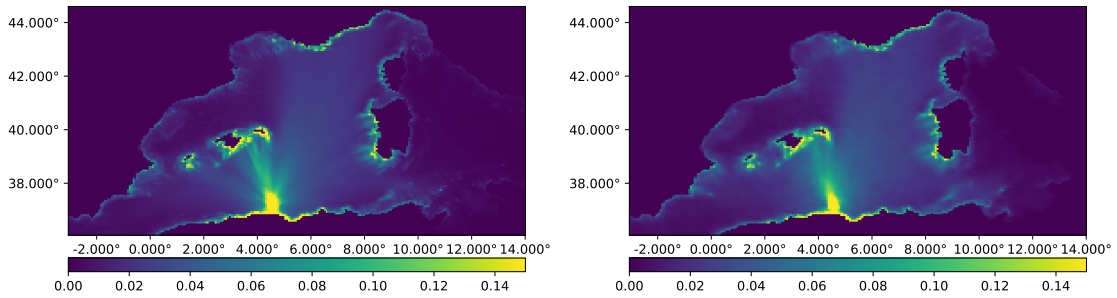
La sélection des hyperparamètres du modèle est encore une fois un problème de taille. On procède à un ajustement manuel des paramètres en étudiant sommairement leur influence sur les erreurs de validation obtenues. On observe qu'un nombre important d'étapes de compression (convolutions avec un *stride* de deux) est nécessaire pour obtenir des résultats intéressants. On obtient ainsi un modèle à six niveaux de compression. On note qu'augmenter le nombre de couches de convolution supplémentaires par niveau ou augmenter la taille de noyau ne semble pas améliorer les performances du réseau, mais augmente significativement le temps de calcul. On retient ainsi une taille de noyau de trois, et un nombre de convolutions par niveau de deux. Le nombre total de couches de convolution est donc de 33, pour un nombre de paramètres entraînables légèrement supérieur à 11 millions, ce qui reste bien en dessous des nombres de paramètres contenus dans les réseaux étudiés précédemment. En effet, pour un MLP produisant des grilles de sortie 256×256 et dont la dernière couche cachée contient 1000 neurones, le nombre de paramètres associés uniquement à la couche de sortie dépasse déjà les 67 millions.



(a) Hauteurs cibles (m)

(b) Hauteurs prédites (m)

FIGURE 38 – Hauteurs maximales (m) simulées et prédites dans les grilles mères pour un scénario de magnitude 6.4 en zone 6

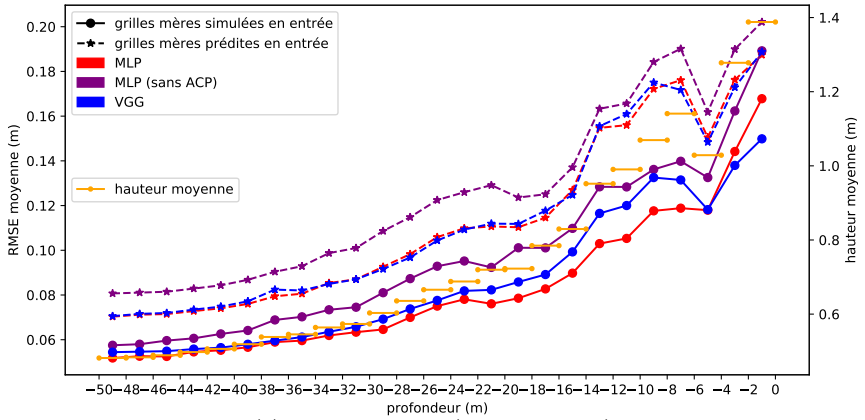


(a) Hauteurs cibles (m)

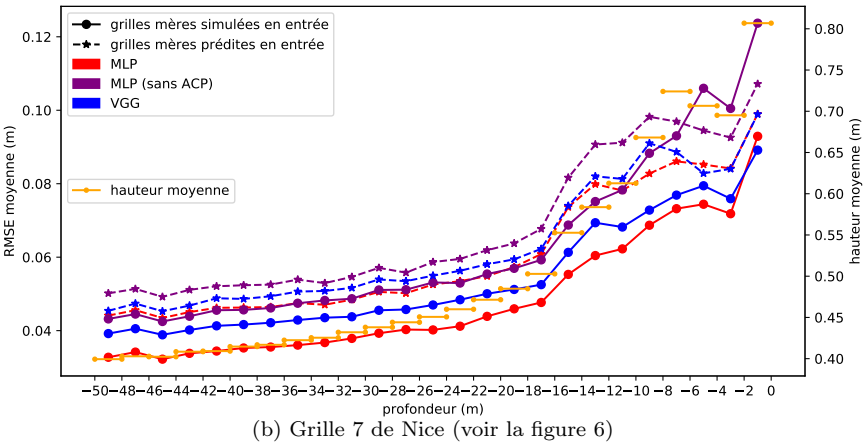
(b) Hauteurs prédites (m)

FIGURE 39 – Hauteurs maximales (m) simulées et prédites dans les grilles mères pour un scénario de magnitude 6.8 en zone 3

On limitait auparavant le surapprentissage par l'ajout de *dropout* au niveau des couches denses du réseau. Ce modèle ne contenant que des couches de convolution, on utilise cette fois des couches de *spatial dropout* [26]. Le *dropout* classique désactive aléatoirement la sortie d'un neurone durant l'entraînement. Le *spatial dropout* désactive un canal (filtre) entier de l'image de sortie d'une couche. La logique derrière cette méthode est que deux pixels voisins au sein d'un filtre issu d'une convolution sont très corrélés. Dans le cas d'un *dropout* classique, cette corrélation n'est pas considérée et la contribution d'un pixel au gradient peut être ignorée tandis que celle de son voisin est prise en compte. Le *spatial dropout* s'assure que toutes les contributions d'un même filtre soient simultanément prises en compte ou ignorées.



(a) Grille 5 de Nice (voir la figure 6)



(b) Grille 7 de Nice (voir la figure 6)

FIGURE 40 – Comparaison des erreurs moyennes selon la profondeur pour les scénarios à forte amplitude

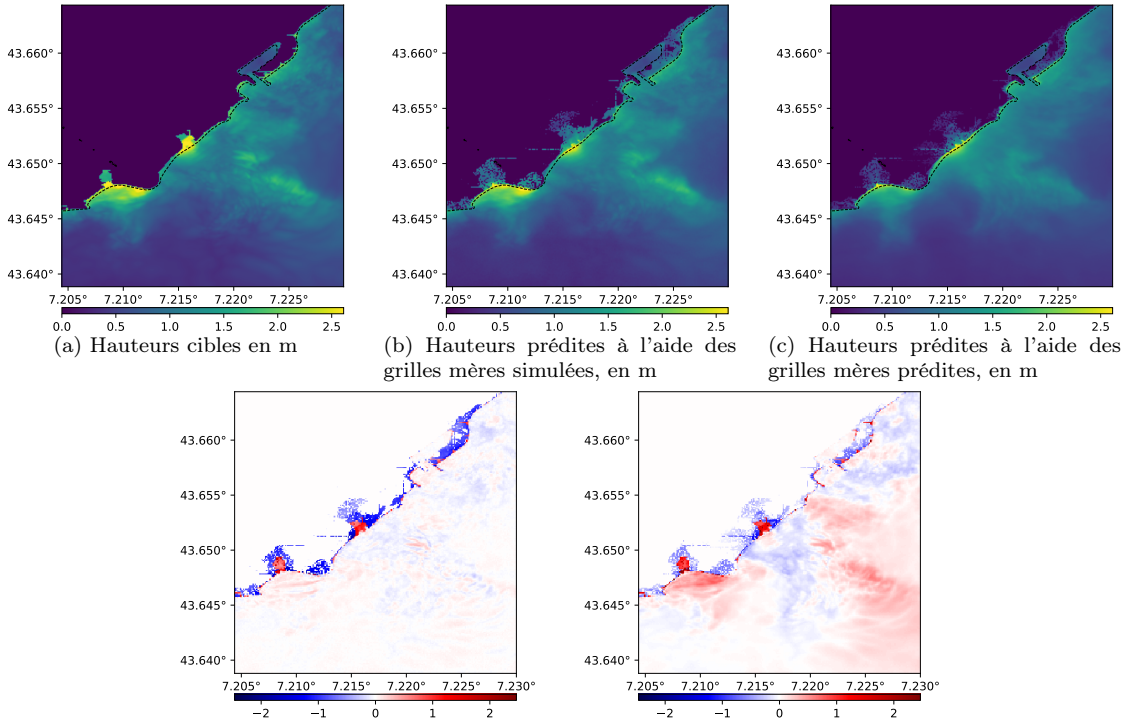
Les figures 38 et 39 donnent des exemples de prédictions de grilles mères. Les hauteurs d'eau prédites sont globalement fidèles aux hauteurs attendues. L'atténuation des hauteurs de vagues causée par un obstacle (ici les Baléares) semble bien retranscrite (figure 39), mais la direction de propagation du Tsunami ne présente pas le même degré de précision (figure 39). La figure 52 en annexe présente un autre exemple de prédiction, et la figure 53 donne les grilles de conditions initiales utilisées pour effectuer ces trois prédictions.

7.2 Modèle complet

Une fois la grille mère prédite, on peut utiliser l'un des réseaux introduits précédemment pour effectuer une prédiction sur l'une des sous-grilles découpée de la grille fille. On peut donc constituer un modèle complet en combinant le modèle prédisant les grilles mères, qui prend les conditions initiales en entrée, et un modèle prédisant les grilles filles en prenant en entrée les grilles mères sous-résolues prédites par le premier réseau. On obtient alors un modèle capable d'effectuer des prédictions haute résolution en une seconde une fois les paramètres de source (nécessaires à la constitution des conditions initiales) évalués.

Les grilles mères prédites sont fortement sous-résolues et de dimensions 128×256. Contrairement aux modèles précédents on va donc utiliser la grille mère complète en entrée du réseau. La figure 40 compare les erreurs obtenues sur les grilles filles 5 et 7 en utilisant les grilles mères prédites en entrée à celles trouvées en utilisant la sous-grille 3 (voir figure 6) tronquée des grilles mères simulées. On note une détérioration (attendue) des prédictions, mais également un comportement très similaire des réseaux MLP avec ACP et VGG. La différence avec les modèles précédents est très marquée sur les scénarios à forte amplitude, avec plus de $4cm$ d'écart d'erreur sur la grille 5 à certaines profondeurs. Cette différence est moins notable pour les scénarios à moyennes et basses amplitudes, où l'écart passe sous le centimètre. Les erreurs d'inondations sont similaires à celles obtenues précédemment.

Les figures 41 (ci-dessous) et 55 (en annexe) présentent deux exemples de prédictions et les comparent aux prédictions obtenues à partir des grilles mères simulées. La figure 54 en annexe donne les tracés hmax/hmax à l'isobathe $1m$ sur l'ensemble des sous-grilles de sortie pour ces deux scénarios.



(a) Hauteurs cibles en m

(b) Hauteurs prédites à l'aide des grilles mères simulées, en m

(c) Hauteurs prédites à l'aide des grilles mères prédites, en m

(d) Erreur signée (grilles mères simulées), en m

(e) Erreur signée (grilles mères prédites), en m

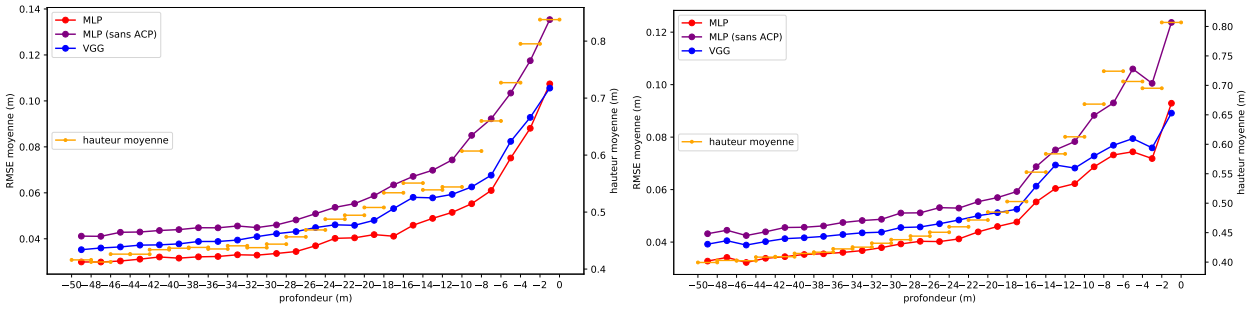
FIGURE 41 – Grilles 5 (voir la figure 6) de Nice simulées et prédites pour un scénario de magnitude 6.8 en zone 5 (MLP avec ACP)

8 Évaluation approfondie d'un modèle sur Nice et Antibes

Les études effectuées dans les sections précédentes permettent d'obtenir une première idée du comportement des modèles de réseaux de neurones considérés. L'objectif de cette partie est de se concentrer sur un modèle et de l'évaluer de manière plus exhaustive. Le jeu de données test de Nice, non utilisé auparavant, ainsi que le jeu de validation d'Antibes seront mis à contribution dans cette analyse.

8.1 Choix du modèle

Les tracés d'erreurs des parties précédentes fournissent un aperçu des différences de performances entre les modèles VGG et MLP. Le MLP avec ACP en entrée présente des erreurs plus basses en profondeur, tandis que les erreurs d'inondations sont souvent inférieures pour le VGG. La figure 42 compare les erreurs obtenues selon la profondeur sur le jeu de validation de Nice pour les grilles 2 (port de Nice) et 7 (Saint-Laurent-du-Var), en considérant uniquement les scénarios à forte amplitude. La figure 56 en annexe présente les mêmes tracés pour les grilles restantes. On note que le VGG présente de légers artefacts de surapprentissage sur certaines grilles (quelques cellules noires). Pour éviter des pics d'erreurs aux profondeurs correspondantes, les hauteurs de vague au niveau des cellules noires situées en profondeurs ont été lissées en faisant la moyenne des hauteurs voisines.



(a) Grille 3 de Nice (voir la figure 6)

(b) Grille 7 de Nice (voir la figure 6)

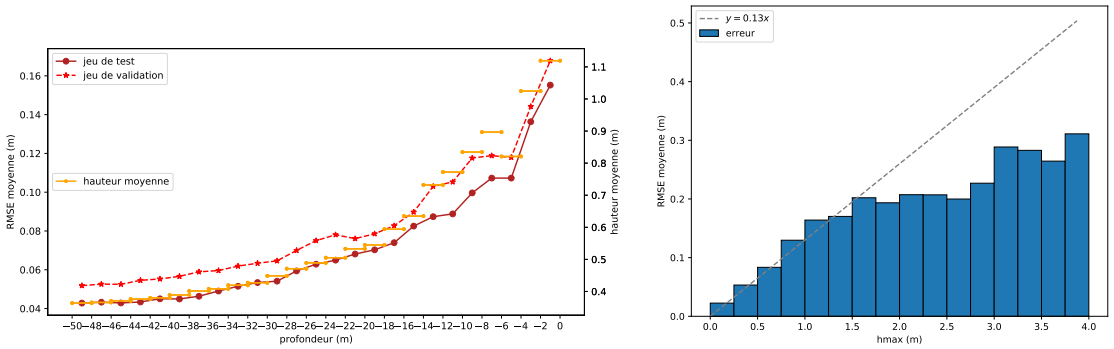
FIGURE 42 – Comparaison des erreurs par tranche de profondeur sur le jeu de validation de Nice (scénarios de hautes amplitudes)

Ces tracés d'erreurs confirment le comportement observé sur la grille 5 (aéroport) : en profondeur le réseau le plus performant est généralement le MLP avec une ACP en entrée, suivi par le VGG, puis le MLP sans ACP. Les différences entre les trois modèles sont souvent faibles. La grille 1 (Villefranche-sur-Mer) est la seule à favoriser le MLP sans ACP devant le VGG, mais avec un écart d'erreur très faible. On retrouve le même type de tracés en considérant les scénarios à moyennes et basses amplitudes. Les erreurs aux niveau des inondations sont également semblables à celles observées sur la grille 5, le VGG présentant généralement des erreurs plus basses que le MLP avec ACP. Cependant, les prédictions d'inondations restent dans chaque cas assez peu précises. Les contributions aux erreurs d'inondations sont concentrées sur quelques scénarios extrêmes, et la localisation des cellules inondées par les réseaux est très approximative. Il est difficile de juger si cette différence d'erreur est réellement due à une meilleure capacité du réseau VGG à prédire les inondations.

Le modèle MLP est donc plus précis en profondeur, et chaque modèle est assez imprécis au niveau des inondations. On note aussi que le VGG est davantage porté au surapprentissage. Le MLP avec ACP paraissant en l'état plus fiable, on sélectionne ce dernier.

8.2 Évaluation du modèle

Cette section se concentre sur l'évaluation du MLP avec ACP en entrée, qui est le modèle sélectionné en partie précédente. Les tracés d'erreurs précédents se basent sur le jeu de données de validation. Le jeu de données de test de Nice n'a pas encore été exploité, et est utilisé en dernier recours pour l'évaluation du modèle sélectionné. Le jeu de données d'Antibes, généré avec les mêmes scénarios que celui de Cannes, n'a pas



(a) Comparaison des erreurs moyennes par tranche de profondeur pour les jeux de test et de validation (m)

(b) Erreur moyenne en fonction de hmax entre $-2m$ et $0m$ pour le jeu test (m)

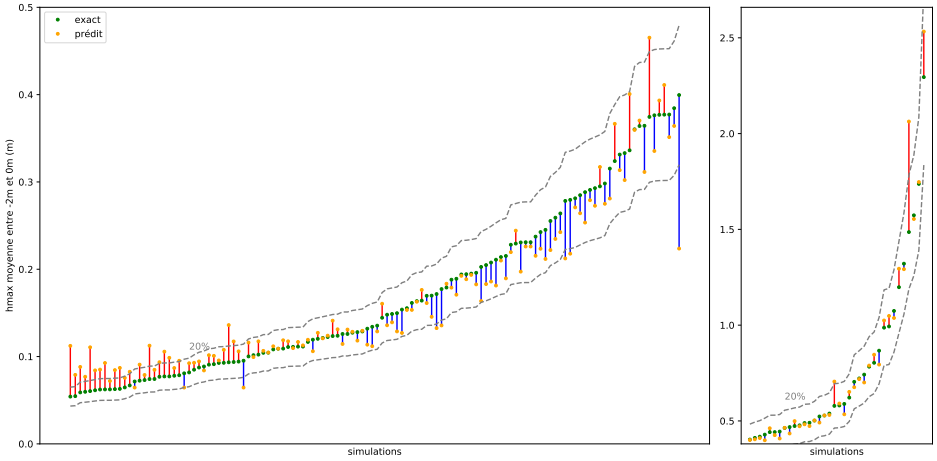
FIGURE 43 – Erreurs obtenues sur la grille 5 de Nice (figure 6) pour les scénarios à forte amplitude. Les hauteurs moyennes sont calculées sur le jeu de test.

non plus été utilisé pour la sélection du modèle. On peut donc raisonnablement recourir au jeu de validation d'Antibes pour évaluer le MLP et comparer les résultats obtenus à ceux trouvés sur Nice, en prenant en compte les différences entre les deux sites.

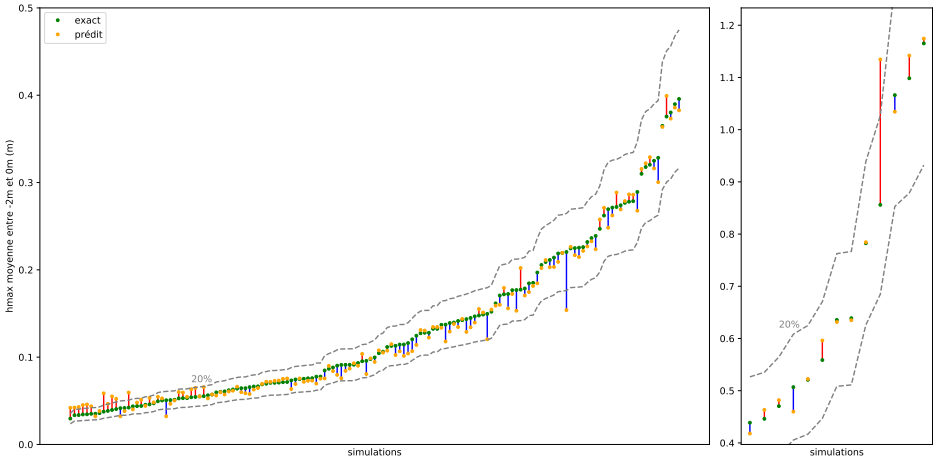
On peut commencer par tracer l'erreur moyenne selon la profondeur sur chaque grille de Nice, de sorte à comparer avec les erreurs obtenues sur le jeu de validation. Les erreurs trouvées sur les deux jeux de données devraient être très similaires sur une même grille. La figure 43 représente les erreurs obtenues sur la grille 5 (aéroport). Les résultats trouvés sur le jeu de test sont cohérents avec ceux trouvés sur le jeu de validation, les erreurs étant même légèrement inférieures pour les scénarios à forte amplitude. Le jeu de validation contient davantage de scénarios très extrêmes, ce qui explique cet écart d'erreur. La différence est moins marquée sur certaines grilles où les hauteurs de vagues sont réduites, ainsi que pour les scénarios présentant des amplitudes de vagues moyennes et faibles. La figure 43b donne le tracé de l'erreur moyenne en fonction de la hauteur maximale de vague pour les hautes amplitudes. On note une erreur restant globalement sous les 15%. La figure 57 en annexe présente les tracés d'erreur selon la profondeur pour les grilles 2 (port de Nice) et 7 (Saint-Laurent-du-Var). On retrouve des erreurs faibles dans les zones immergées. Les erreurs au niveau des inondations sont également semblables à celles obtenues auparavant, et les prédictions d'inondations restent donc imprécises pour les fortes amplitudes.

Les prédictions semblent satisfaisantes en moyenne en profondeur. En évaluant l'ensemble des scénarios à proximité de la côte (figure 44), on note des différences assez significatives entre hauteur d'eau simulée et hauteur prédite pour quelques scénarios, en particulier sur la grille 5. L'écart entre les deux valeurs reste généralement inférieur à 20%.

On peut également comparer les grilles simulées et prédites pour quelques simulations présentant des erreurs importantes. La figure 45 présente un tel tracé sur la grille 5 (aéroport) ainsi que les erreurs en profondeur sur chaque grille. Les autres grilles sont représentées en annexe en figure 58. On observe une tendance des réseaux à sur-prédire la hauteur de vague en profondeur mais à sous-estimer les inondations sur ce scénario précis (particulièrement visible sur les grilles 5 et 7). Les prédictions obtenues restent assez fiables en profondeur si l'on prend en compte le fait qu'il s'agît d'un scénario extrême présentant une erreur supérieure à la moyenne.

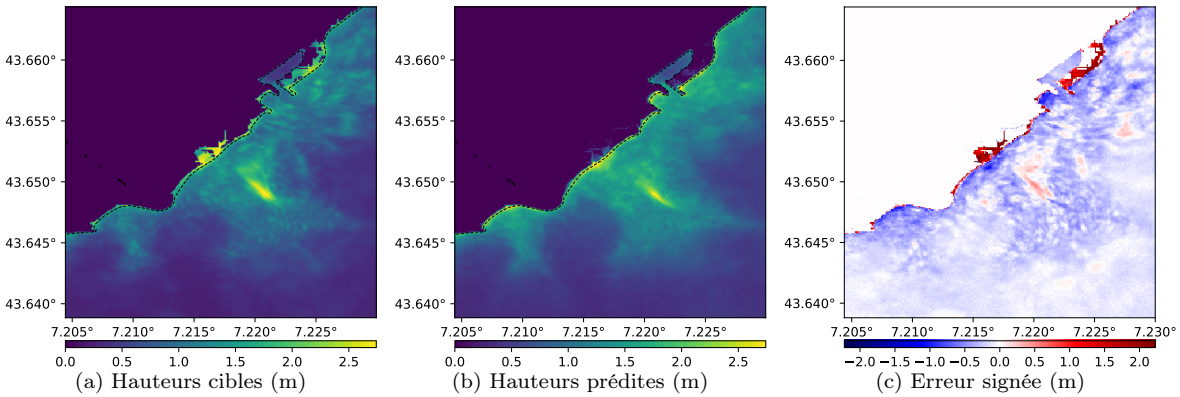


(a) Grille 5



(b) Grille 7

FIGURE 44 – Comparaison des hmax moyens entre $-2m$ et $0m$ sur le jeu de test de Nice

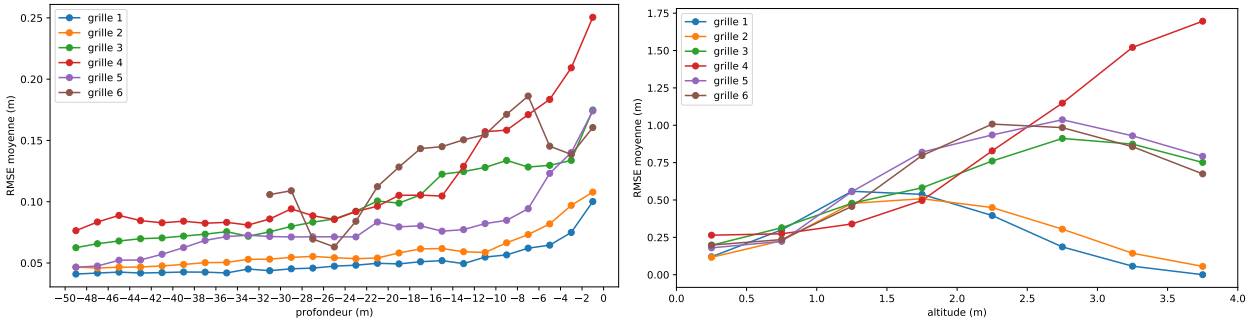


(a) Hauteurs cibles (m) (b) Hauteurs prédites (m) (c) Erreur signée (m)

Grille	1	2	3	4	5	6	7
Erreur (m)	0.20253743	0.20105096	0.16826121	0.21062715	0.32398025	0.19914141	0.23499046
Hauteur moyenne (m)	0.637	0.649	0.706	0.847	1.086	0.614	0.696

(d) Erreurs RMSE et hauteurs moyennes simulées entre -20 et $0m$ de profondeur (m)

FIGURE 45 – Grilles simulées et prédites sur la grille 5 de Nice pour un scénario de magnitude 6.9 en zone 5



(a) Erreur moyenne par tranche de profondeur sur Antibes (m) (b) Erreur moyenne par tranche d'altitude sur Antibes (m)

Grille (figure 51)	1	2	3	4	5	6
Hauteur moyenne (m)	1.039	1.104	1.637	2.140	1.334	1.535

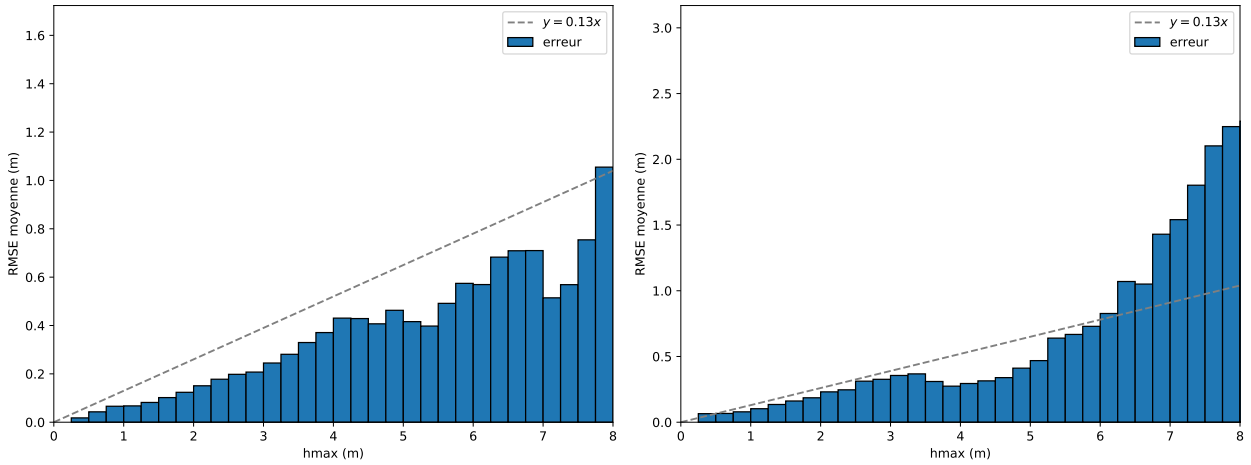
(c) Hauteurs moyennes simulées entre -20 et $0m$ de profondeur (m)

FIGURE 46 – Comparaison des erreurs sur le jeu de validation d'Antibes (scénarios de hautes amplitudes)

Enfin, on peut étudier le comportement du MLP sur le jeu de validation d'Antibes, non exploité jusqu'à présent. La figure 46 présente les erreurs moyennes selon la profondeur obtenues sur l'ensemble des grilles, calculées en considérant uniquement les scénarios de validation à forte amplitude. On remarque des erreurs assez importantes sur certaines grilles, qui s'expliquent par des hauteurs d'eau très élevées dans les simulations (voir la figure 46c).

La figure 47 donne le tracé de l'erreur en fonction de la hauteur de vague sur les grilles 4 (criques de la Garoupe) et 6 (Juan-les-Pins) pour les scénarios à forte amplitude, la grille 4 présentant les vagues les plus importantes ainsi que des inondations massives. Les simulations atteignent des hauteurs anormales sur certaines cellules (jusqu'à plus de $10m$), qui sont probablement dues à des erreurs numériques ou des phénomènes d'amplification dans les simulations. Ce comportement aberrant est reproduit par les prédictions des réseaux de neurones, les scénarios d'entraînement présentant les mêmes anomalies. On note tout de même une explosion de l'erreur sur la grille 6 pour les hauteurs supérieures à $6m$, qui est sûrement causé par le comportement imprévisible de ces anomalies.

La figure 48 donne la répartition des erreurs pour chaque tranche de profondeur sur l'ensemble des scénarios pour la grille 4. On note que plus de la moitié des prédictions présentent une erreur entre $-2m$



(a) Grille 4 (b) Grille 6

FIGURE 47 – Erreur moyenne en fonction de Hmax entre $-2m$ et $0m$ pour les scénarios à forte amplitude sur le jeu de validation d'Antibes (m)

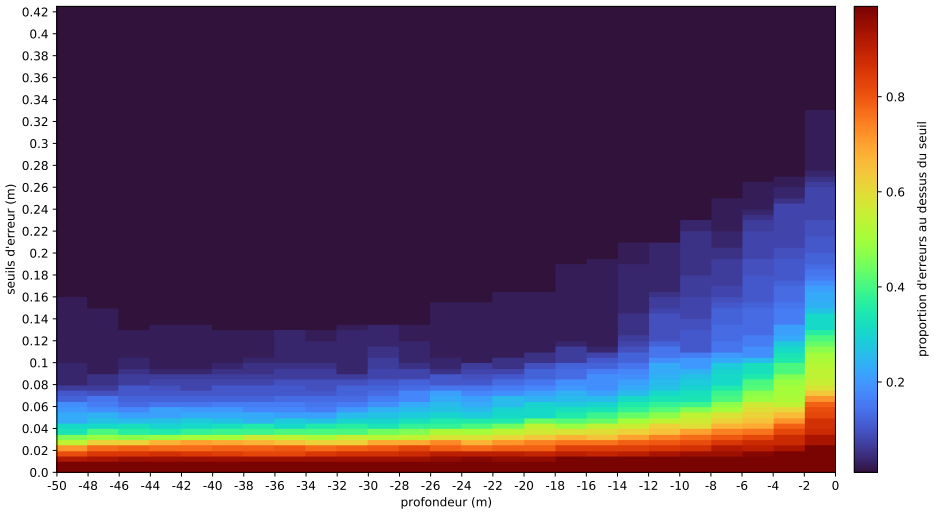


FIGURE 48 – Distribution des RMSE sur l'ensemble des scénarios pour chaque tranche de profondeur de la grille 4 d'Antibes

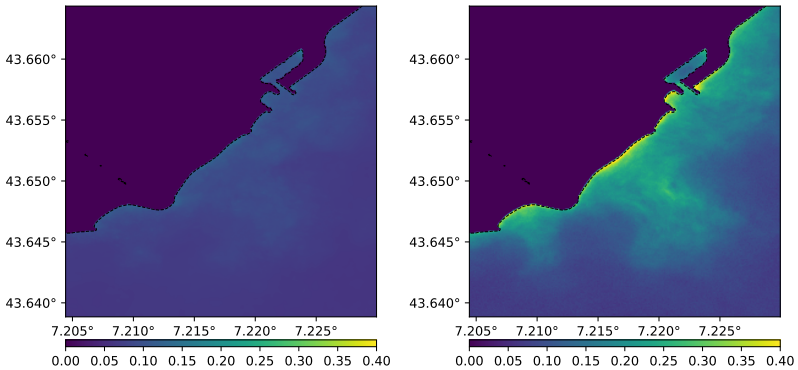
et $0m$ de profondeur supérieure à $10cm$, ce qui s'explique par les fortes hauteurs de vagues observées. La figure 59 en annexe présente un exemple de prédiction sur quelques grilles de sortie d'Antibes. Les hauteurs de vagues restent en général inférieures à $2m$ pour ce scénario, mais les inondations sont très importantes et sont assez bien localisées dans les prédictions.

Les résultats obtenus sur le jeu de test de Nice et le jeu de validation d'Antibes semblent globalement se conformer aux observations précédentes, avec des erreurs relatives similaires à celles trouvées sur les jeux de validation de Cannes et Nice.

8.3 Généralité du modèle

Une interrogation demeure sur la capacité du modèle sélectionné à se généraliser à des situations éloignées de celles rencontrées durant l'apprentissage. Les paramètres de sources choisis pour les scénarios d'entraînement et de validation appartiennent à la même base de faille, et les calculs d'erreur montrent que l'apprentissage est efficace au sein de cet espace de paramètres. Les prédictions sont cependant très détériorées lorsque les paramètres de la simulation sont trop différents de ceux vus durant l'entraînement.

La figure 49 montre un exemple de prédiction sur Nice pour un tsunami causé par un séisme historique



(a) Hauteurs cibles (m) (b) Hauteurs prédites (m)

FIGURE 49 – Grilles simulées et prédites sur la grille 5 de Nice pour un scénario de magnitude 8.5 issu de la zone de subduction hellénique

issu de la zone de subduction hellénique [6], qui est absente des scénarios d'entraînement. La simulation est de plus effectuée sur un temps plus long afin de permettre aux vagues de se propager jusqu'à Nice. Les hauteurs de vague sont très largement sur-prédites sur la majorité des grilles (figure 50). Cette faiblesse doit être prise en compte lors l'utilisation des modèles.

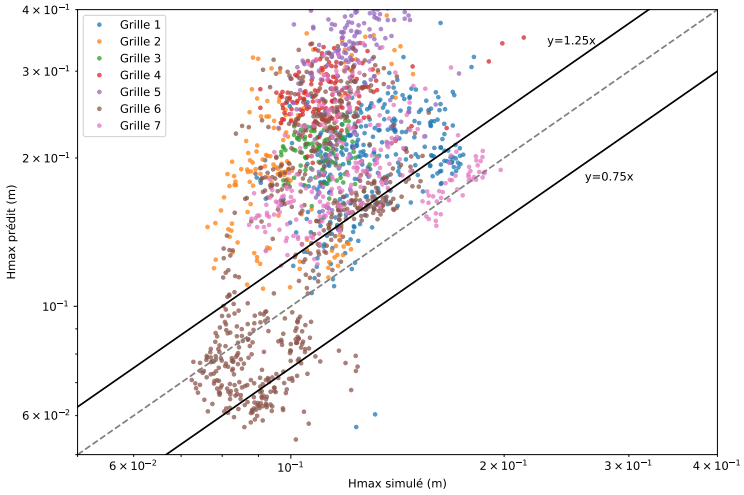


FIGURE 50 – Hauteurs prédites en fonction des hauteurs simulées autour de l'isobathe $1m$ pour un scénario de magnitude 8.5 issu de la zone de subduction hellénique

Conclusion

Ce stage a permis de mettre en valeur les atouts et les inconvénients de l'utilisation de réseaux de neurones pour la prévision rapide de hauteurs de vagues et d'inondations de tsunamis en haute résolution.

Dans un premier temps, un jeu de données centré sur la ville de Cannes et généré lors d'un stage précédent [1] a été exploité afin d'étudier et de paramétrer un modèle MLP capable de prédire des grilles haute résolution à partir de grilles grossières (section 3). L'ajout d'une analyse en composantes principales en entrée a entraîné une réduction des erreurs observées. Une recherche d'hyperparamètres optimaux a rendu possible la sélection d'un modèle performant.

À la suite de cette étude, d'autres architectures de réseaux à couches de convolution ont été considérées sur une seconde base de données plus conséquente et centrée sur Nice (section 5). L'influence de la taille de la base d'apprentissage a été sommairement étudiée (section 5.4), ainsi que la possibilité de transférer l'apprentissage d'un réseau d'une zone de prédiction à une autre, les premiers résultats présentés restant peu concluants dans ce domaine. Les tracés d'erreurs semblent cependant suggérer qu'il est possible de limiter la quantité de simulations d'entraînement nécessaires en les choisissant de manière plus judicieuse.

L'évaluation des modèles VGG et MLP a montré de grandes similarités entre les deux réseaux, le modèle MLP avec ACP en entrée se montrant légèrement plus efficace sur les cas de validation. Les mêmes types de résultats ont été observés pour les deux modèles, soit une bonne précision dans les prédictions en profondeur mais des prédictions d'inondations souvent approximatives pour les scénarios extrêmes. Ce comportement a été confirmé dans une évaluation plus approfondie du MLP sur les sites de Nice et d'Antibes (section 8.2).

Les prédictions d'inondations sont donc encore peu satisfaisantes pour les scénarios à très fortes hauteurs de vagues, ce qui constitue une limite importante pour l'application en contexte opérationnel. Les différentes études effectuées ont mis en exergue la forte dépendance des modèles aux données d'entraînement. La variété des données fournies a un impact important sur la capacité des modèles à se généraliser à de nouveaux scénarios, et les prédictions issues de scénarios trop éloignés de l'espace des simulations d'entraînement sont peu fiables. De plus, les réseaux de neurones ainsi entraînés n'ont pas vocation à dépasser les codes de simulation mais à émuler leurs résultats. Les anomalies ou imprécisions rencontrées durant l'apprentissage seront donc reproduites au moment de l'inférence. On note qu'il serait intéressant d'améliorer la précision des simulations haute résolution fournies à l'apprentissage, quitte à augmenter considérablement la durée de calcul de la base d'entraînement. En effet, ces calculs étant effectués une unique fois en amont, cela aurait un impact positif sur la qualité des prédictions sans accroître le temps d'inférence.

Malgré ces limites de poids, l'approche IA reste intéressante en contexte d'alerte. Les réseaux sont capables de discriminer les scénarios à forte ou faible amplitude, et de fournir une première estimation des hauteurs de vagues en profondeur généralement assez précise (erreur généralement inférieure à 25%) en moins d'une minute, en prenant en compte le temps de calcul de la grille mère. L'étape d'inférence par les réseaux de neurones est peu coûteuse et dure moins d'une seconde sur un ordinateur de bureau. La section 7 propose un modèle capable d'effectuer des prédictions à partir des conditions initiales d'un scénario, sans demander la simulation de la grille grossière par un supercalculateur. Le temps d'estimation total est donc réduit à moins d'une seconde, au prix d'un surcoût d'erreur, et la capacité de calcul nécessaire est largement diminuée. Enfin, on peut noter que les modèles considérés durant l'étude produisent des prédictions sur des zones géographiques assez réduites (de l'ordre de quelques kilomètres carrés). Il serait cependant possible d'élargir considérablement la zone de prédiction en se restreignant à des profondeurs d'intérêt (par exemple au dessus de $50m$) sans modifier les architectures out les méthodes de calcul établies.

De nombreuses voies restent à explorer, que ce soit au niveau de la génération des données ou des modèles de prédiction utilisés. Une première piste serait de mettre en place un algorithme efficace de sélection des simulations d'entraînement selon les paramètres de sources des séismes considérés. Une telle sélection permettrait de grandement réduire le coût de calcul de la base d'entraînement, ainsi que d'augmenter la qualité de l'apprentissage conséquent. Un autre enjeu important est la prise en compte des équations différentielles et de la bathymétrie dans l'apprentissage des réseaux de neurones. Dans le cadre d'un modèle effectuant des prédictions en espace et en temps, ces contraintes physiques pourraient être incluses dans l'entraînement par l'ajout d'un terme de pénalisation. La dérivation de la sortie du réseau selon ses entrées permet en effet dans ce cas de mesurer la conformité du modèle aux équations [22]. Enfin, une dernière idée est de développer un modèle capable d'effectuer des prédictions en haute résolution sur l'ensemble de la côte méditerranéenne française, la zone de prédiction étant spécifiée au moment de l'inférence. Une possibilité serait d'entraîner un

réseau effectuant des prédictions à résolution croissante en cascade, en se basant pour l'entraînement sur les grilles intermédiaires utilisées lors de la simulation. Les réseaux de convolution de type GAN (*Generative Adversarial Networks*) semblent appropriés pour une telle application. Ces réseaux étant notoirement difficiles à entraîner, la constitution d'une base d'apprentissage robuste en sera un enjeu d'autant plus crucial.

Références

[1] Audrey Chouly. Perceptron multicouche pour l'estimation rapide des inondations de tsunami : application au site côtier méditerranéen de cannes, October 2020.

[2] Joël Clément and Dominique Reymond. New Tsunami Forecast Tools for the French Polynesia Tsunami Warning System. *Pure and Applied Geophysics*, 172(3-4) :791–804, March 2015.

[3] Burcu Erkmen and Tülay Yildırım. Improving classification performance of sonar targets by applying general regression neural network with pca. *Expert Systems with Applications*, 35(1) :472–475, 2008.

[4] Ardiansyah Fauzi and Norimi Mizutani. Machine learning algorithms for real-time tsunami inundation forecasting : a case study in nankai region. *Pure and Applied Geophysics*, pages 1–14, 2019.

[5] Audrey Gailler, Hélène Hébert, François Schindelé, and Dominique Reymond. Coastal Amplification Laws for the French Tsunami Warning Center : Numerical Modeling and Fast Estimate of Tsunami Wave Heights Along the French Riviera. *Pure and Applied Geophysics*, 175(4) :1429–1444, April 2018.

[6] Audrey Gailler, F Schindelé, and H Hébert. Impact of hellenic arc tsunamis on corsica (france). *Pure and Applied Geophysics*, 173(12) :3847–3862, 2016.

[7] Fei He and Lingying Zhang. Prediction model of end-point phosphorus content in bof steelmaking process based on pca and bp neural network. *Journal of Process Control*, 66 :51–58, 2018.

[8] P. Heinrich, A. Jamelot, A. Cauquis, and A. Gailler. Taitoko, an advanced code for tsunami propagation, developed at the french tsunami warning centers. *European Journal of Mechanics - B/Fluids*, 88 :72–88, 2021.

[9] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012.

[10] Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. Introvae : introspective variational autoencoders for photographic image synthesis. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 52–63, 2018.

[11] Haris Iqbal. Harisiqbal88/plotneuralnet v1.0.0. `https://github.com/HarisIqbal88/PlotNeuralNet`, December 2018.

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv e-prints*, page arXiv :1611.07004, November 2016.

[13] T. Jayalakshmi and A. Santhakumaran. Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3(1) :1793–8201, 2011.

[14] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2nd ed. edition, 2010.

[15] Diederik P. Kingma and Jimmy Ba. Adam : A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv :1412.6980, December 2014.

[16] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband : A Novel Bandit-Based Approach to Hyperparameter Optimization. *arXiv e-prints*, page arXiv :1603.06560, March 2016.

[17] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, volume 2013, pages 436–440, 2013.

[18] Dominic Masters and Carlo Luschi. Revisiting Small Batch Training for Deep Neural Networks. *arXiv e-prints*, page arXiv :1804.07612, April 2018.

[19] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J. Kennedy. Relational autoencoder for feature extraction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 364–371, 2017.

[20] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net : Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *arXiv e-prints*, page arXiv :1606.04797, June 2016.

[21] Iyan E Mulia, Aditya Riadi Gusman, and Kenji Satake. Applying a deep learning algorithm to tsunami inundation database of megathrust earthquakes. *Journal of Geophysical Research : Solid Earth*, 125(9) :e2020JB019690, 2020.

[22] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks : A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378 :686–707, 2019.

[23] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, page arXiv :1409.1556, September 2014.

[24] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *arXiv e-prints*, page arXiv :1206.2944, June 2012.

[25] Viviane Souty and Audrey Gailler. Tsunami hazard associated to earthquakes along the French coasts. A probabilistic approach (PTHA). In *EGU General Assembly Conference Abstracts*, EGU General Assembly Conference Abstracts, page 5554, May 2020.

[26] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient Object Localization Using Convolutional Networks. *arXiv e-prints*, page arXiv :1411.4280, November 2014.

[27] Ahmad Zia Ul-Saufie, Ahmad Shukri Yahaya, Nor Azam Ramli, Norrimi Rosaida, and Hazrul Abdul Hamid. Future daily pm10 concentrations prediction by combining regression models and feedforward backpropagation models with principle component analysis (pca). *Atmospheric Environment*, 77 :621–630, 2013.

[28] Peter J. Verveer and Robert P. W. Duin. An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on pattern analysis and machine intelligence*, 17(1) :81–86, 1995.

[29] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184 :232–242, 2016. RoLoD : Robust Local Descriptors for Computer Vision 2014.

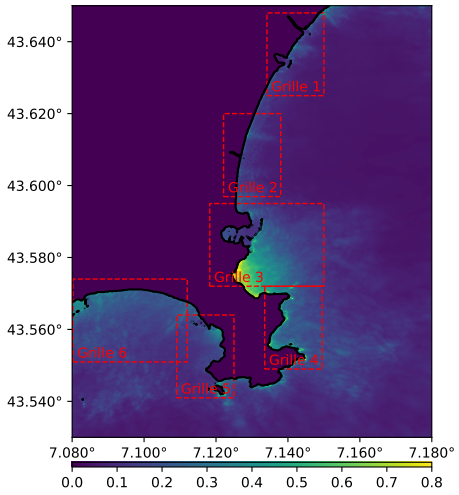
A Sous-grilles mères et filles

Grille	Longitude min	Latitude min	Longitude max	Latitude max	Taille totale	Taille tronquée
Fille 1	7.005	43.525	7.045	43.552	108000	74320
Fille 2	6.9531	43.535	7.005	43.552	88230	27738
Fille 3	7.04	43.535	7.07	43.57	105000	42282
Mère 1	4	39	9	44	22500	19198
Mère 2	1	36.5	10.25	44.45	65330	49095

TABLE 7 – Paramètres des grilles mères et filles pour la ville de Cannes (grilles mères à $3700m$)

Grille	Long. min	Lat. min	Long. max	Lat. max	Nombre de cellules
Fille 1	7.302	43.694	7.328	43.707	32768
Fille 2	67.269	43.686	7.295	43.699	32768
Fille 3	7.244	43.684	7.269	43.697	32768
Fille 4	7.228	43.664	7.241	43.69	32768
Fille 5	7.204	43.639	7.23	43.6644	65536
Fille 6	7.192	43.6374	7.204	43.663	32768
Fille 7	7.166	43.6462	7.192	43.659	32768
Mère 1	6.967	43.233	7.5	43.77	4096
Mère 2	6.683	42.733	7.75	43.8	16384
Mère 3	6.025	42.875	8.162	43.945	32768
Mère 4	6.025	41.808	8.162	43.945	65536
Mère 5	3.892	39.675	8.162	43.945	65536

TABLE 8 – Paramètres des grilles mères et filles pour la ville de Nice (grilles mères à $900m$). La grille mère 5 est redimensionnée par interpolation bilinéaire.



(a) Visualisation des grilles filles

Grille	Long. min	Lat. min	Long. max	Lat. max
Fille 1	7.134	43.6245	7.15	43.648
Fille 2	7.122	43.597	7.138	43.62
Fille 3	7.118	43.572	7.15	43.595
Fille 4	7.134	43.549	7.150	43.572
Fille 5	7.109	43.541	7.125	43.564
Fille 6	7.080	43.551	7.112	43.574

(b) Paramètres des grilles filles

FIGURE 51 – Sous-grilles filles considérées sur Antibes

B Sélection des paramètres du MLP

Réduction	Couches cachées	Neurones par couches	Taux d'apprentissage	MSE sur le jeu de validation
Non	1	1024	0.000 100	0.008 123
Non	1	1024	0.000 138	0.008 268
50 (ACP)	2	704	0.000 465	0.007 292
50 (ACP)	1	864	0.000 204	0.007 322
50 (ACP)	1	704	0.000 109	0.007 389
100 (ACP)	1	1024	0.000 227	0.006 236
100 (ACP)	2	1024	0.000 100	0.006 340
100 (ACP)	2	1024	0.000 139	0.006 490
125 (ACP)	1	960	0.000 471	0.006 177
125 (ACP)	1	960	0.000 209	0.006 263
125 (ACP)	2	704	0.000 410	0.006 489

Table 9 – Meilleurs résultats des recherches bayésiennes à 50 itérations

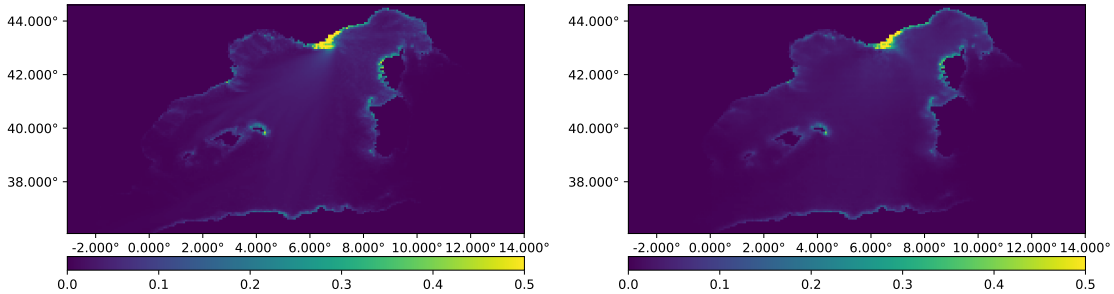
Couches	Neurones	LR	Réduction	Grille 1, 6-*fold CV*	Grille 1	Grille 2	Grille 3
1	1024	$1 \cdot 10^{-4}$	Non	0.07459	0.09249	0.04579	0.07681
1	1024	$1.37666 \cdot 10^{-4}$	Non	0.0722	0.09578	0.04635	0.07353
2	512	$1 \cdot 10^{-3}$	Non	0.08743	0.10994	0.05906	0.09094
2	704	$4.64983 \cdot 10^{-4}$	50 (PCA)	0.06658	0.08195	0.04268	0.0687
1	864	$2.03761 \cdot 10^{-4}$	50 (PCA)	0.06878	0.08302	0.04294	0.07066
2	512	$1 \cdot 10^{-3}$	50 (PCA)	0.06775	0.0916	0.04412	0.07227
2	704	$4.64983 \cdot 10^{-4}$	50 (AE)	0.07033	0.08847	0.04933	0.07601
1	864	$2.03761 \cdot 10^{-4}$	50 (AE)	0.07515	0.10775	0.05317	0.09272
2	512	$1 \cdot 10^{-3}$	50 (AE)	0.07038	0.09563	0.04992	0.08527
1	1024	$2.27403 \cdot 10^{-4}$	100 (PCA)	0.06439	0.07904	0.04195	0.06365
2	1024	$1 \cdot 10^{-4}$	100 (PCA)	0.06561	0.08082	0.04777	0.06807
2	512	$1 \cdot 10^{-3}$	100 (PCA)	0.06655	0.08679	0.04158	0.06807
1	1024	$2.27403 \cdot 10^{-4}$	100 (AE)	0.07225	0.09334	0.05026	0.07407
2	1024	$1 \cdot 10^{-4}$	100 (AE)	0.07221	0.09077	0.05173	0.07471
2	512	$1 \cdot 10^{-3}$	100 (AE)	0.07008	0.08902	0.04687	0.07166
1	960	$4.71383 \cdot 10^{-4}$	125 (PCA)	0.06498	0.07914	0.0394	0.06491
1	960	$2.08517 \cdot 10^{-4}$	125 (PCA)	0.06532	0.07944	0.03975	0.06339
2	512	$1 \cdot 10^{-3}$	125 (PCA)	0.06617	0.08384	0.04432	0.06856
1	960	$4.71383 \cdot 10^{-4}$	125 (AE)	0.06901	0.09053	0.05281	0.07194
1	960	$2.08517 \cdot 10^{-4}$	125 (AE)	0.07107	0.09253	0.05605	0.08161
2	512	$1 \cdot 10^{-3}$	125 (AE)	0.06937	0.08877	0.04872	0.07583

Table 10 – Erreurs RMSE (m), MLP, grille d'entrée reserrée. 6-*fold CV* donne la moyenne des RMSE sur les 6 entraînements.

Couches	Neurones	LR	Réduction	Grille 1, 6-*fold CV*
1	1024	$1 \cdot 10^{-4}$	Non	0.07725
1	1024	$1.37666 \cdot 10^{-4}$	Non	0.07716
2	512	$1 \cdot 10^{-3}$	Non	0.08917
2	704	$4.64983 \cdot 10^{-4}$	50 (ACP)	0.06826
1	864	$2.03761 \cdot 10^{-4}$	50 (ACP)	0.07215
2	512	$1 \cdot 10^{-3}$	50 (ACP)	0.06855
1	1024	$2.27403 \cdot 10^{-4}$	100 (ACP)	0.06628
2	1024	$1 \cdot 10^{-4}$	100 (ACP)	0.06509
2	512	$1 \cdot 10^{-3}$	100 (ACP)	0.06696
1	960	$4.71383 \cdot 10^{-4}$	125 (ACP)	0.06566
1	960	$2.08517 \cdot 10^{-4}$	125 (ACP)	0.06631
2	512	$1 \cdot 10^{-3}$	125 (ACP)	0.06801

TABLE 11 – Erreurs de cross-validation, MLP avec grille d'entrée large (Cannes)

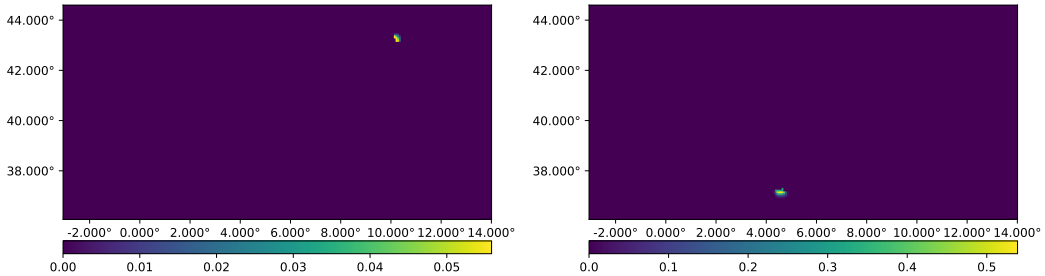
C Prédiction de grilles mères



(a) Hauteurs cibles (m)

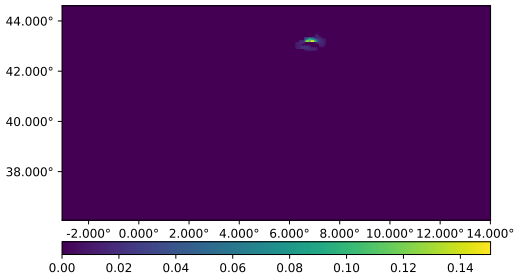
(b) Hauteurs prédites (m)

FIGURE 52 – Hauteurs maximales (m) simulées et prédites dans les grilles mères pour un scénario de magnitude 6.9 en zone 5



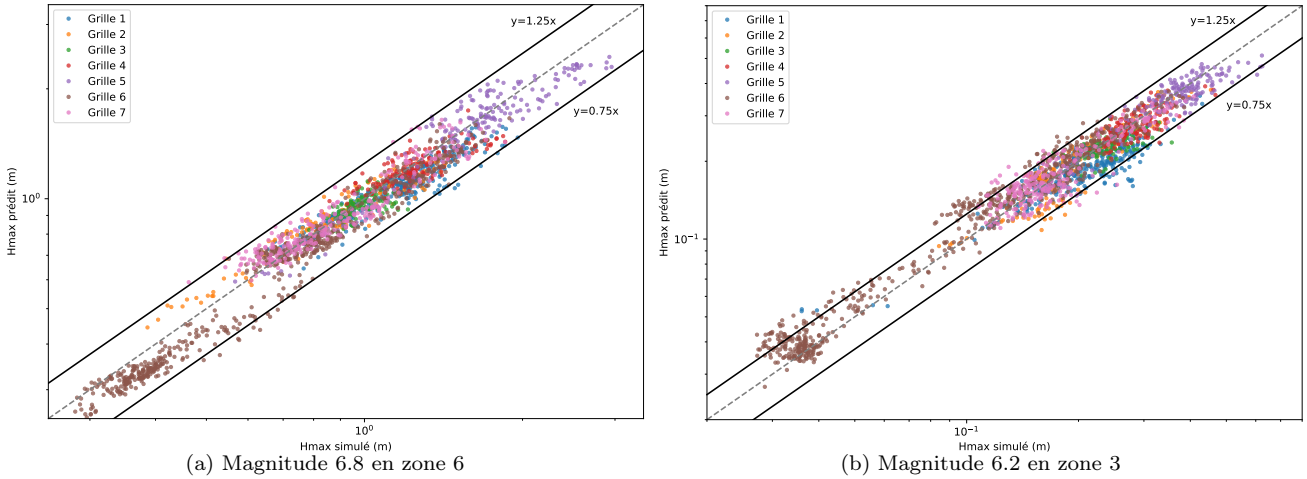
(a) Magnitude 6.4 en zone 6

(b) Magnitude 6.8 en zone 3



(c) Magnitude 6.9 en zone 5

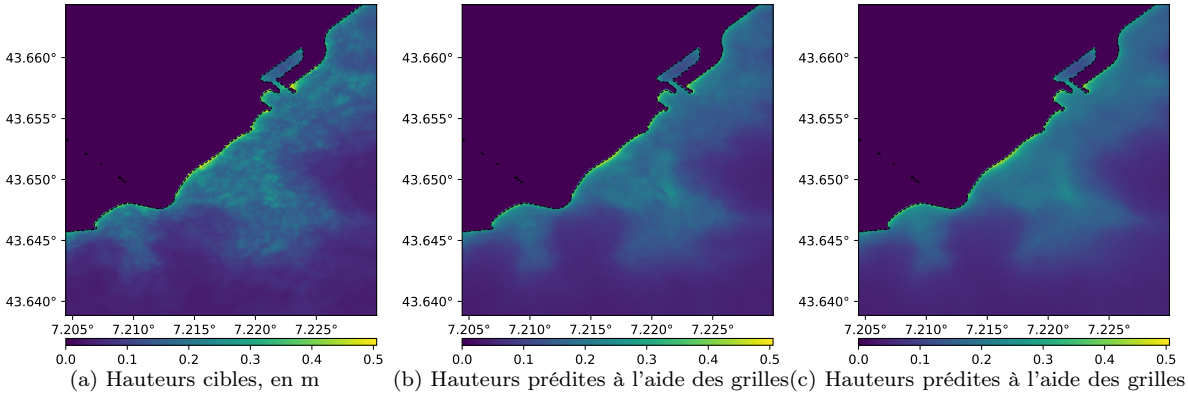
FIGURE 53 – Conditions initiales utilisées pour les prédictions des figures 38, 39, 52



(a) Magnitude 6.8 en zone 6

(b) Magnitude 6.2 en zone 3

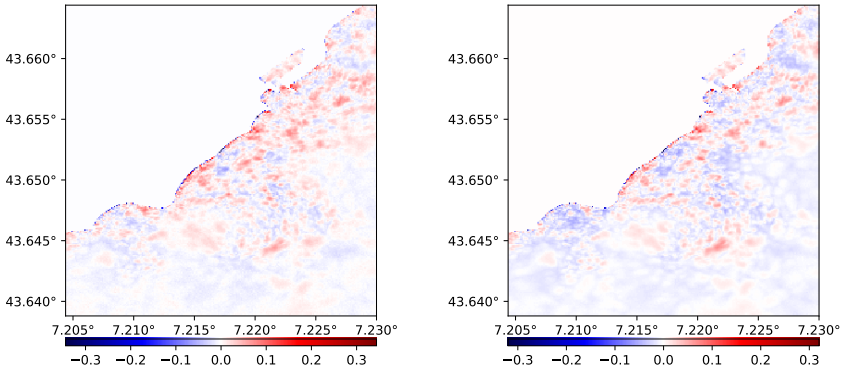
FIGURE 54 – Tracés hmax/hmax des prédictions des scénarios des figures 41 et 55. Le modèle est un MLP avec ACP prenant en entrée les grilles mères prédites par un Vnet à l'aide des conditions initiales.



(a) Hauteurs cibles, en m

(b) Hauteurs prédites à l'aide des grilles mères simulées, en m

(c) Hauteurs prédites à l'aide des grilles mères prédites, en m

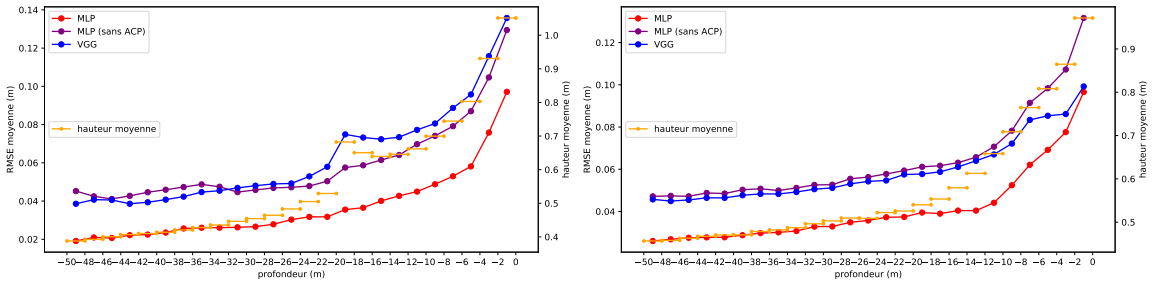


(d) Erreur signée (grilles mères simulées), en m

(e) Erreur signée (grilles mères prédites), en m

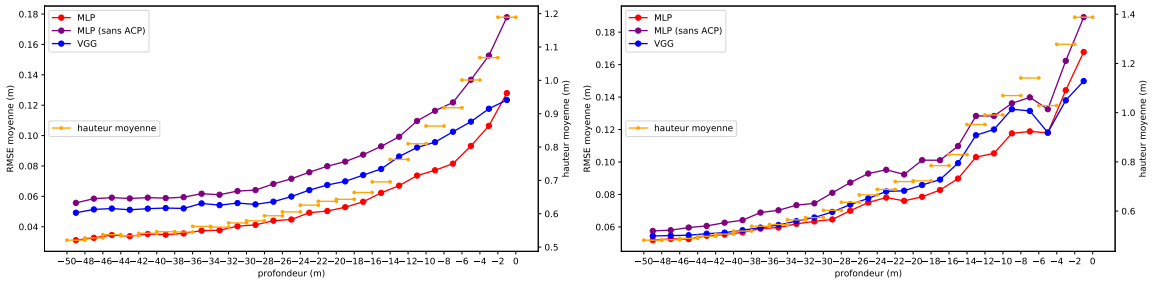
FIGURE 55 – Grilles 5 de Nice simulées et prédites pour un scénario de magnitude 6.2 en zone 5 (MLP avec ACP)

D Graphiques d'évaluation



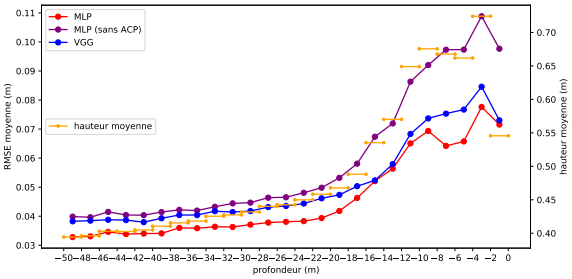
(a) Erreur moyenne par tranche de profondeur, grille 1 de Nice

(b) Erreur moyenne par tranche de profondeur, grille 3 de Nice



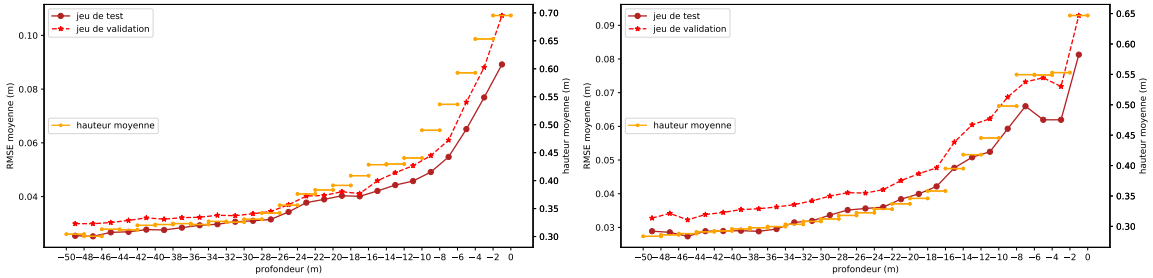
(c) Erreur moyenne par tranche de profondeur, grille 4 de Nice

(d) Erreur moyenne par tranche de profondeur, grille 5 de Nice



(e) Erreur moyenne par tranche de profondeur, grille 6 de Nice

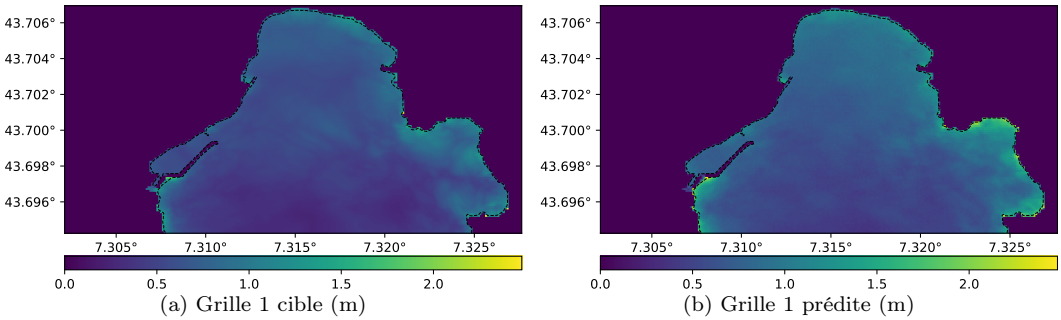
FIGURE 56 – Comparaison des erreurs selon la profondeur sur le jeu de validation de Nice (scénarios de hautes amplitudes)



(a) Erreurs moyennes selon la profondeur sur la grille 2 (m)

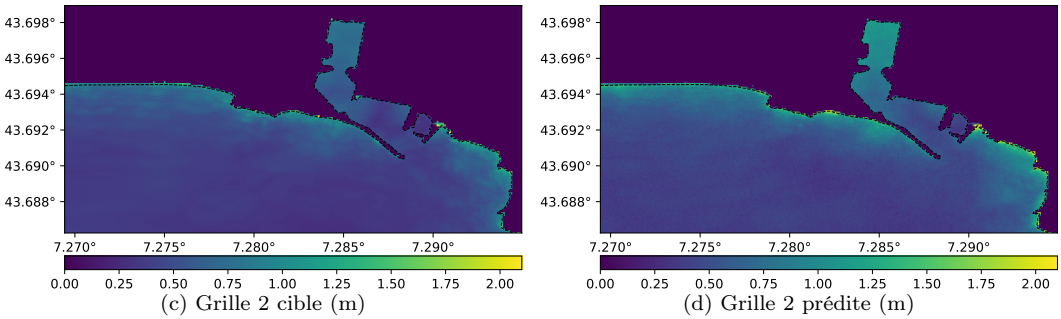
(b) Erreurs moyennes selon la profondeur sur la grille 7 (m)

FIGURE 57 – Comparaison des erreurs moyennes de test et de validation par tranche de profondeur pour les scénarios à forte amplitude de Nice



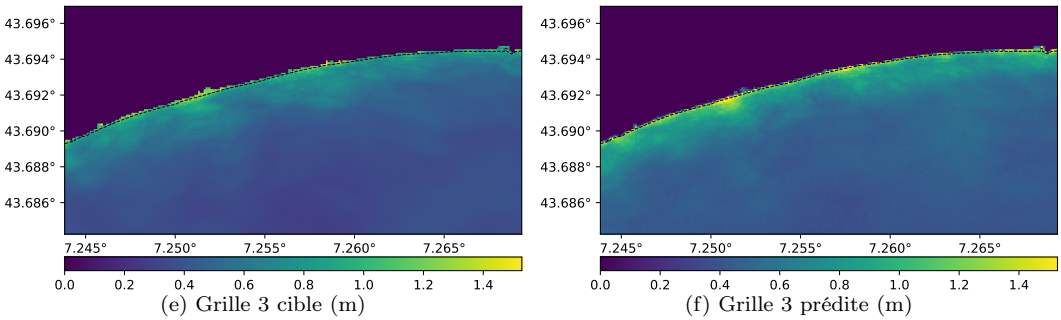
(a) Grille 1 cible (m)

(b) Grille 1 prédite (m)



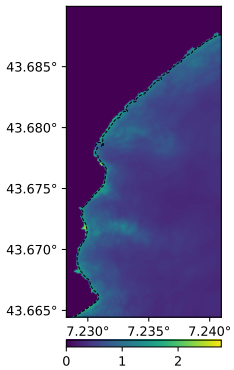
(c) Grille 2 cible (m)

(d) Grille 2 prédite (m)

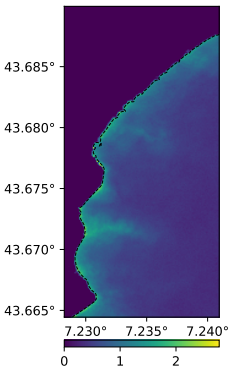


(e) Grille 3 cible (m)

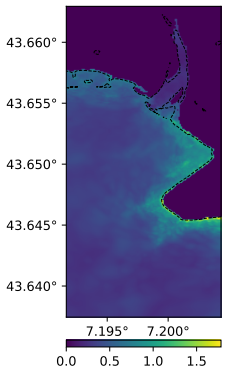
(f) Grille 3 prédite (m)



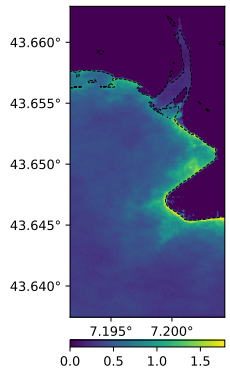
(g) Grille 4 cible (m)



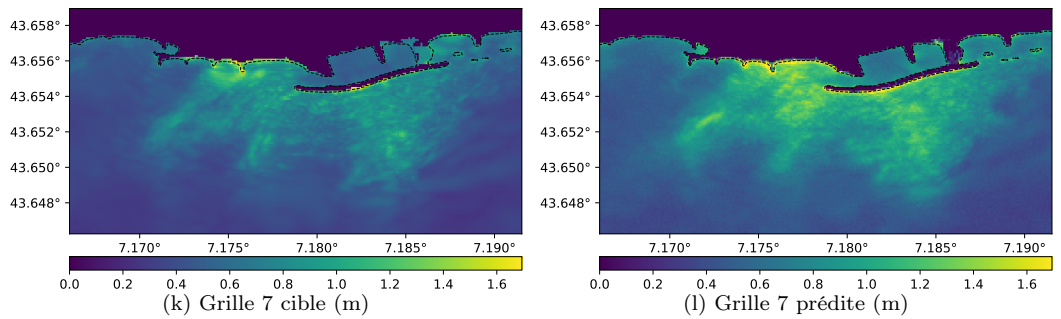
(h) Grille 4 prédite (m)



(i) Grille 6 cible (m)



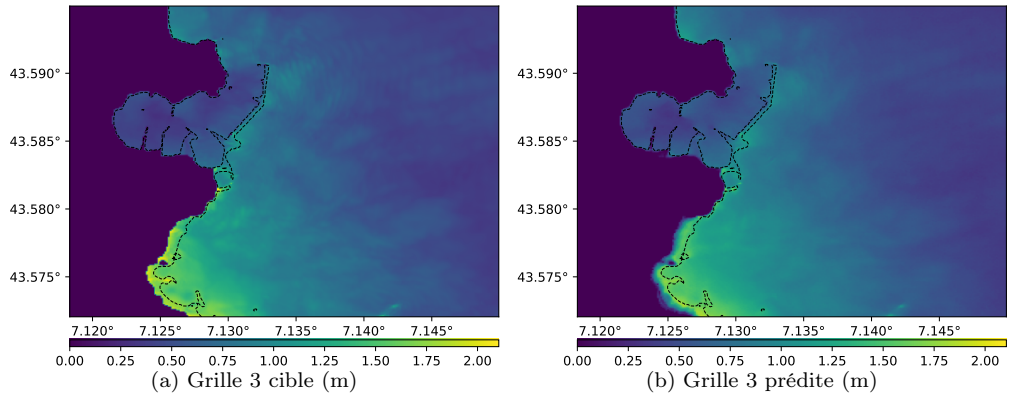
(j) Grille 6 prédite (m)



(k) Grille 7 cible (m)

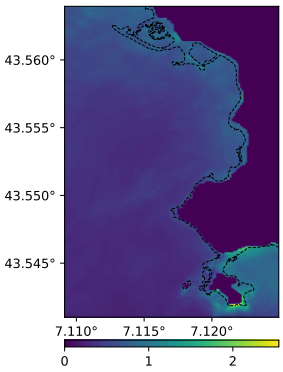
(l) Grille 7 prédite (m)

FIGURE 58 – Grilles simulées et prédites pour un scénario de magnitude 6.9 en zone 5 sur Nice

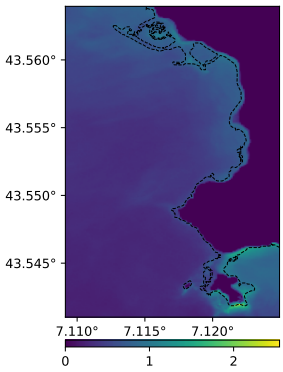


(a) Grille 3 cible (m)

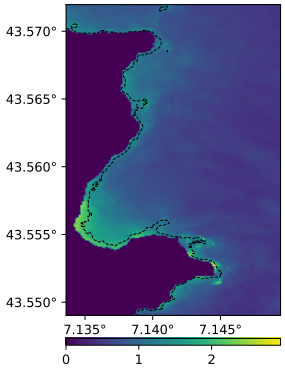
(b) Grille 3 prédite (m)



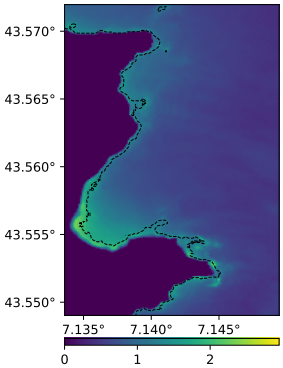
(c) Grille 5 cible (m)



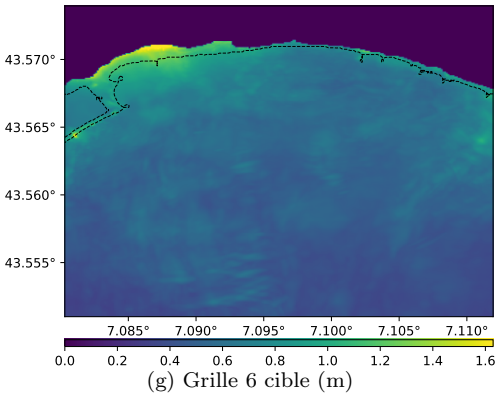
(d) Grille 5 prédite (m)



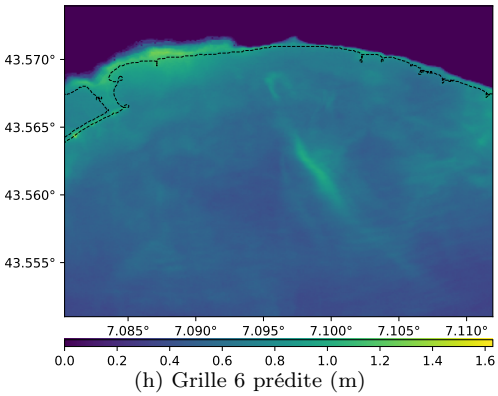
(e) Grille 4 cible (m)



(f) Grille 4 prédite (m)



(g) Grille 6 cible (m)



(h) Grille 6 prédite (m)

FIGURE 59 – Grilles simulées et prédites pour un scénario de magnitude 6.9 en zone 5 sur Antibes