Université de Haute-Alsace

Faculté des Sciences et Techniques (FST)

Master 2 Informatique et Mobilité

Projet Final : Apprentissage Profond

Étude des architectures pour la prédiction d'événements sismologiques majeurs

Auteur :
Oreste MUHIRWA GABO
orestegabo@icloud.com

Enseignant :
M. Maxime DEVANNE
Responsable Deep Learning

16 Janvier 2025

Table des matières

1 Introduction

Ce rapport présente une étude comparative d'architectures de réseaux de neurones profonds appliquées à la détection d'événements sismologiques majeurs. Le centre sismologique nécessite une automatisation capable de traiter plusieurs relevés par minute. En utilisant le jeu de données *Earthquakes* (archive UCR), nous analysons des séries temporelles de 512 heures pour distinguer les signaux "normaux" des signes précurseurs d'un séisme.

2 Méthodologie et Justification des Choix

2.1 Prétraitement et Régularisation

La robustesse de l'évaluation repose sur une normalisation **Z-Score** systématisée pour chaque série temporelle. Cette étape est cruciale car elle permet au réseau de se focaliser sur les variations de fréquences et de formes d'ondes plutôt que sur l'amplitude brute du signal. Pour contrer le surapprentissage, nous avons utilisé :

- **Dropout (0.3) :** Utilisé dans le classifieur final pour désactiver aléatoirement des neurones, forçant le modèle à apprendre des représentations redondantes.
- **Batch Normalization :** Appliquée dans le CNN pour stabiliser le gradient et accélérer la convergence.
- **Weighted CrossEntropy Loss :** Un poids de 3.0 est attribué à la classe minoritaire pour pallier le déséquilibre (74.8% de cas normaux).

2.2 Optimisation et Reproductibilité

Nous utilisons l'optimiseur **Adam** couplé au scheduler **OneCycleLR**. Ce dernier est justifié par sa capacité à effectuer une "super-convergence" en faisant varier le taux d'apprentissage de manière cyclique. Pour garantir la reproductibilité, une graine aléatoire fixe (*Seed 42*) a été utilisée, bien que nous observions des variations mineures dues aux opérations non-déterministes de l'accélération matérielle MPS.

3 Analyse des Architectures

3.1 Multi-Layer Perceptron (MLP)

Le MLP sert de base de référence (Baseline). Il traite les 512 points comme un vecteur plat. **Analyse des logs :** Avec une perte finale de 0.0001 mais une précision de test stagnante à 69.78%, le modèle montre un surapprentissage clair. Il mémorise le bruit du train set sans généraliser la structure temporelle.

3.2 Convolutional Neural Network (CNN)

Le CNN utilise des noyaux 1D pour extraire des caractéristiques spatiales locales (pics d'activité). **Analyse des logs :** Bien qu'il atteigne une *Best Accuracy* de 75.54%, il reste instable. Sa matrice de confusion montre qu'il peine à isoler les séismes réels (Rappel de 0.37).

3.3 Recurrent Neural Network (RNN/LSTM)

Le RNN est l'architecture la plus adaptée théoriquement aux séries temporelles grâce à ses cellules à mémoire. **Analyse des logs :** Le RNN s'est révélé être le modèle le plus performant avec une précision record de **81.29%** et un score F1 de **0.54** pour les séismes. Il identifie 21 séismes sur 35, un résultat significatif pour la sécurité sismologique.

3.4 Modèle Hybride : SeismicNet

Fusionnant CNN et Bi-LSTM, ce modèle vise à extraire des motifs locaux avant de les traiter séquentiellement. **Analyse des logs :** Il atteint une précision de 78.42% et se distingue par son temps d'inférence extrêmement optimisé (0.0045s).

4 Étude de Complexité et Performance

Conformément aux contraintes matérielles du centre sismologique, nous présentons le tableau comparatif suivant, exécuté sur architecture MacBook Pro M4 :

Modèle	Paramètres	Inférence (s)	Best Acc.	F1-Séis.
MLP	74,050	0.0067s	74.82%	0.25
CNN	**43,842**	0.0088s	75.54%	0.35
RNN	50,562	0.0074s	**81.29%**	**0.54**
SeismicNet	85,698	**0.0045s**	78.42%	0.37

TABLE 1 – Comparaison technique des modèles entraînés sur 100 époques.

5 Discussion sur la Robustesse et Variabilité

Il est important de noter que l'entraînement de réseaux profonds est un processus stochastique. Bien que la graine soit fixée, des exécutions répétées peuvent mener à des résultats légèrement différents (variations de $\pm 2\%$). Cette variabilité souligne la nécessité d'une évaluation robuste par matrice de confusion. Par exemple, le RNN présente un équilibre sain entre précision (0.49) et rappel (0.60) pour la classe Earthquake, ce qui est crucial pour minimiser les faux négatifs (séismes non détectés).

6 Conclusion

Le modèle **RNN** est recommandé pour sa capacité de détection supérieure (Meilleur Score F1). Toutefois, le **SeismicNet** représente l'avenir du déploiement opérationnel grâce à son temps d'inférence record, permettant de traiter des milliers de relevés par minute sur les machines modestes du service.

7 Annexes : Code et Implémentation

```python
class SeismicNet(nn.Module):
    def __init__(self, input_dim=1, hidden_dim=64):
        super().__init__()
        self.feature_extractor = nn.Sequential(
            nn.Conv1d(input_dim, 32, kernel_size=7, padding=3),
            nn.BatchNorm1d(32), nn.ReLU(), nn.MaxPool1d(2),
            nn.Conv1d(32, 64, kernel_size=5, padding=2),
            nn.BatchNorm1d(64), nn.ReLU(), nn.MaxPool1d(2)
        )
        self.lstm = nn.LSTM(64, hidden_dim, bidirectional=True,
    batch_first=True)
        self.classifier = nn.Sequential(
            nn.Linear(hidden_dim * 2, 64), nn.ReLU(),
            nn.Dropout(0.3), nn.Linear(64, 2)
        )

    def forward(self, x):
        x = x.unsqueeze(1)
        x = self.feature_extractor(x).transpose(1, 2)
        lstm_out, _ = self.lstm(x)
        return self.classifier(lstm_out[:, -1, :])
```

Listing 1 – Architecture du SeismicNet Hybride