UNIVERSITÉ DU QUÉBEC À MONTRÉAL

TRAVAIL PRATIQUE INDIVIDUEL

PRÉSENTÉ À

PHILIPPE GOULET COULOMBE

DANS LE CADRE DU COURS

DONNÉE MASSIVE & APPRENTISSAGE AUTOMATIQUE AVEC APPLICATION EN ÉCONOMIE

ECO930J, Groupe 40

PAR

VAUDESCAL GUILLAUME (VAUG30119904)

DATE DE REMISE

28 AVRIL 2022

Table des matières

Introduction :

Le présent travail a pour but de m'initier à l'utilisation des Transformers sur une série temporelle afin de prédire le rendement d'un actif. Dans ce travail en vue de cette finalité, j'utilise les données du prix de l'action Royal Bank of Canada (RBC) et quelques autres de ses caractéristiques (cours d'ouverture, de haut, de bas, de clôture et du volume de transactions) en fréquence journalière.

Méthodologie :

Cette section présente les outils et méthodes utilisées afin d'obtenir nos résultats. Principalement, je me suis inspiré de la méthodologie de plusieurs articles scientifiques sur le sujet, que je cite tout au long de mon travail.

Données :

L'ensemble de données de Royal Bank of Canada est en fréquence journalière, commençant le 16-10-1995, et se terminant le 21-01-2022. De plus, nous disposons des cours d'ouverture, de haut, de bas, de clôture ainsi que du volume de transactions de l'action RBC.

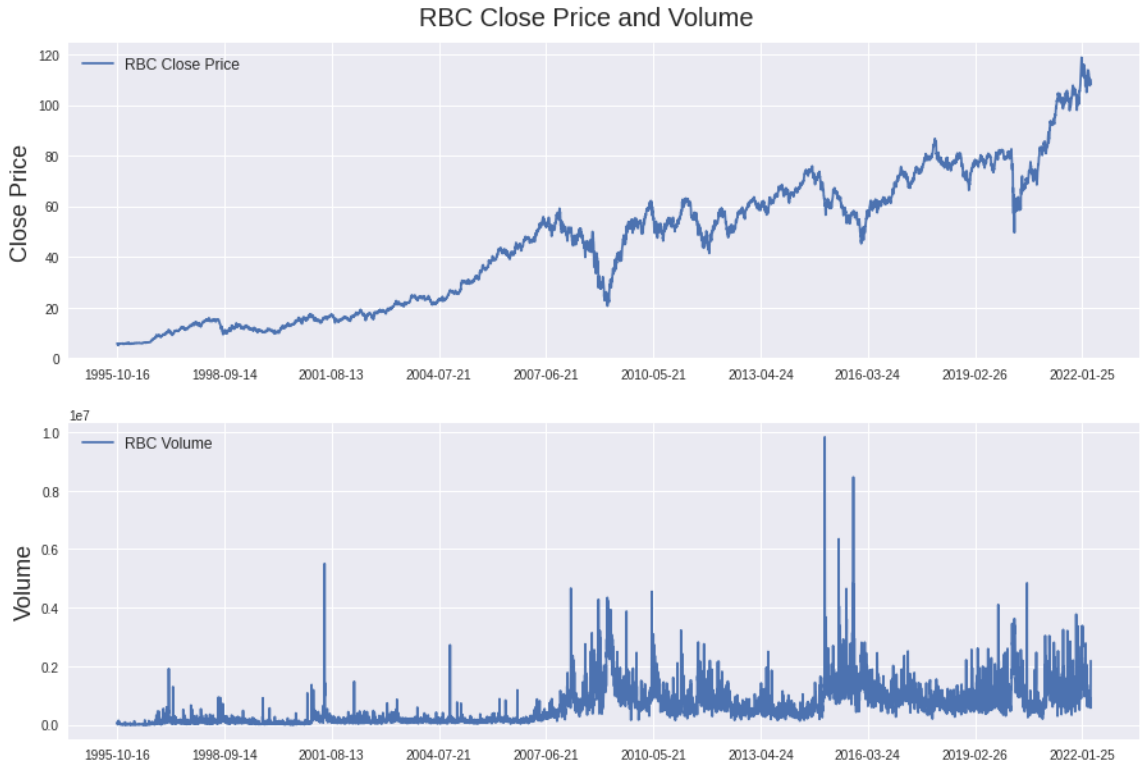

Figure 1 : Prix de clôture et le volume journalier de RBC.

Préparation des données :

Les variables de prix et de volume sont converties en taux de rendements boursiers quotidiens et en variations de volume quotidiennes, ce qui permet d'augmenter la stationnarité de nos données. Également une normalisation min-max est appliquée, et la série chronologique est divisée en un ensemble d'apprentissage, de validation et de test set.

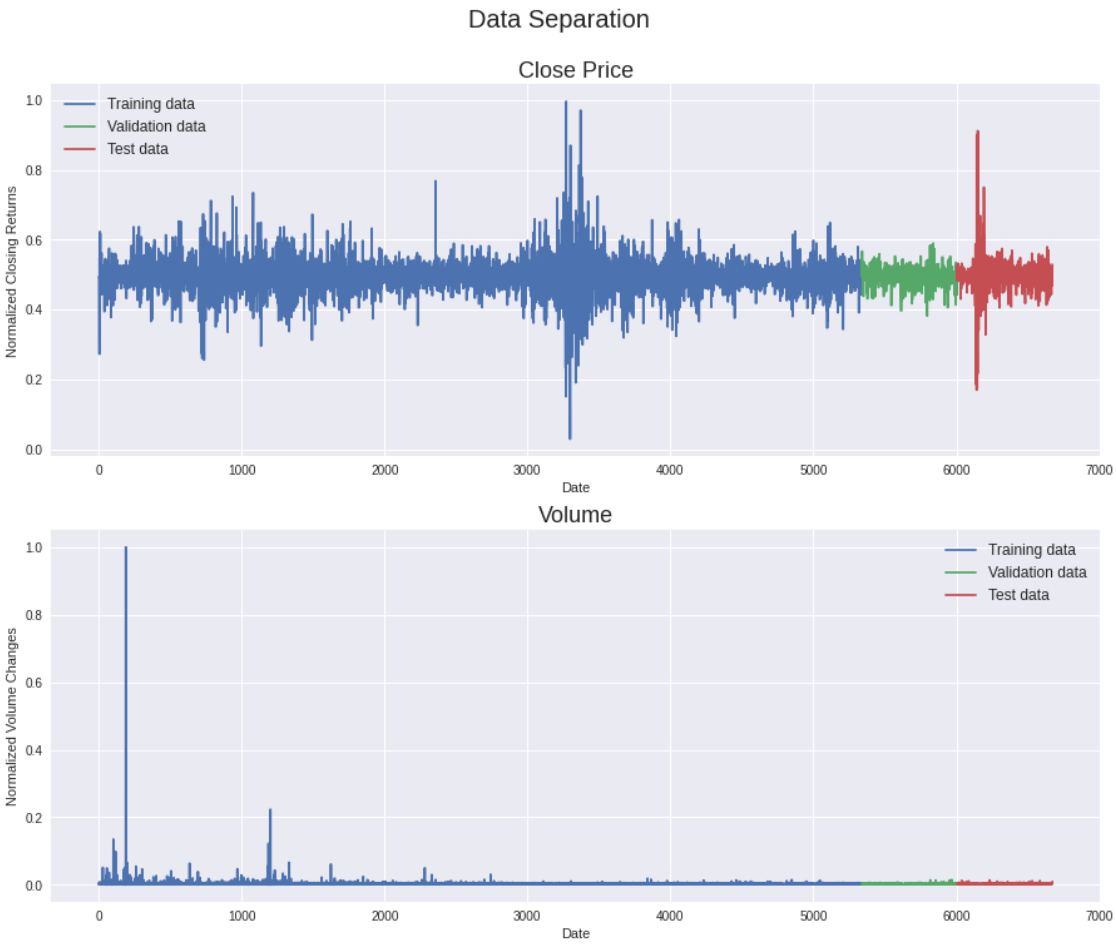

Figure 2 : Séparation des données.

Enfin, les ensembles de *training*, de *validation* et de *test* set, sont séparés en séquences individuelles d'une durée de 128 jours chacune. Pour chaque jour de séquence, les 4 caractéristiques de prix (*Open, High, Low, Close*) et la caractéristique Volume sont présentes, ce qui donne 5 caractéristiques par jour. Au cours d'une seule étape d'apprentissage, notre modèle Transformers recevra en entrée 32 séquences (*batch_size* = 32) d'une durée de 128 jours (*seq_len*=128) et comportant 5 caractéristiques par jour.

$$(32, 128, 5)$$

Figure 3 : Taille de la matrice d'entrée du modèle.

Incorporations temporelles (Times Embeddings) :

La première étape de l'implémentation de notre modèle Transformers est de considérer comment encoder dans notre modèle la notion de temps qui est cachée dans les prix des actions. Lors du traitement de données de séries temporelles, le temps est une caractéristique essentielle. Cependant, lors du traitement de données temporelles avec un Transformers, les séquences sont transmises en une seule fois à travers l'architecture du Transformers ce qui rend difficile l'extraction des dépendances temporelles.

Pour pallier à cela, nous allons rajouter au modèle une notion de temps afin de traiter nos cours de bourse. En ce sens, nous utilisons les *Times Embeddings*[1] qui permettent d'apprendre et classifier les données temporelles. Sans les *Times Embeddings*, notre Transformers ne recevrait aucune information sur l'ordre temporel de nos cours de bourse. Ce qui pourrait donner par exemple, qu'un cours de bourse de 2020 puisse avoir la même influence sur la prédiction du cours de bourse de demain, qu'un cours de bourse de l'année 1990. Ce que nous évitons ainsi par l'utilisation des *Times Embeddings* dans notre modèle.

Time2Vector :

Afin de surmonter les indifférences temporelles d'un Transformer, et en lien avec notre point précèdent sur les *Times Embeddings*, nous allons mettre en œuvre l'approche décrite dans l'article « *Time2Vec : Learning a Vector Representation of Time* »[2]. Les auteurs de cet article, proposent une représentation vectorielle du temps agnostique au modèle, appelée *Time2Vec*. Ce vecteur sera ainsi intégré dans notre architecture du modèle afin d'améliorer les performances de ce dernier. Les principales idées à retenir de l'article que nous allons utiliser sont les deux suivantes ;

[1] Tanya Goyal et Greg Durrett. « *Intégration d'expressions temporelles pour les modèles d'ordre temporel profond.* » 2019, Dans Actes de la 57e réunion annuelle de l'Association pour la linguistique computationnelle, pages 4400–4406, Florence, Italie. Association pour la linguistique computationnelle.

[2] Kazemi, S.M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., & Brubaker, M.A. « *Time2Vec: Learning a Vector Representation of Time.* » 2019, URL : ArXiv, abs/1907.05321.

Premièrement, les auteurs ont identifié qu'une représentation efficace du temps doit inclure les notions de périodicités et non périodicités. Un exemple de modèle périodique peut être par exemple, le montant des ventes d'un magasin, qui est plus élevé les week-ends ou les jours fériés. En revanche, un exemple de modèle non périodique serait une maladie, qui survient avec une forte probabilité, plus le patient est âgé.

Deuxièmement, une représentation temporelle doit être invariante par rapport à la remise à l'échelle du temps, ce qui signifie que la représentation temporelle n'est pas affectée par différents incréments de temps, par exemple (jours, heures ou secondes) et de longs horizons temporels.

En combinant les idées de modèles périodiques et non périodiques ainsi que l'invariance par rapport au changement d'échelle temporelle, nous obtenons la définition mathématique suivante que nous allons détailler :

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$

Figure 4 : Représentation mathématique du vecteur temps.

On peut noter que le vecteur temporel $t2v$ est constitué de deux composantes, où $\omega_i\tau + \varphi_i$ (fonction linéaire) représente la caractéristique non périodique et $F(\omega_i\tau + \varphi_i)$ la caractéristique périodique du vecteur temporel. Également, ω dans $\omega_i\tau + \varphi_i$ est une matrice qui définit la pente de notre série temporelle noté τ , et φ correspond à une matrice qui définit l'endroit où notre série temporelle τ coupe l'axe des y.

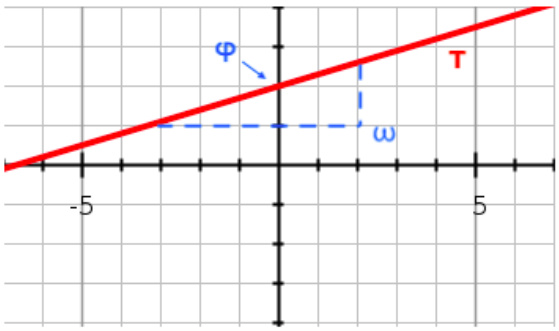

Figure 5 : Représentation 2D d'une caractéristique temporelle non périodique.

La deuxième composante $F(\omega_i\tau + \varphi_i)$ représente la caractéristique périodique du vecteur temps. Comme précédemment, nous avons encore la partie linéaire $\omega_i\tau + \varphi_i$ cependant ici la fonction linéaire est comprise dans une fonction supplémentaire noté $F()$. Les auteurs ont expérimenté différentes fonctions pour décrire au mieux une relation périodique (sigmoïde, ReLU...). Au final, une fonction sinus a obtenu les meilleures performances et les plus stables. En combinant la fonction linéaire $\omega_i\tau + \varphi_i$ avec une fonction sinus, la représentation 2D se présente comme suit : φ décale la fonction sinus le long de l'axe des x et ω détermine la longueur d'onde de la fonction sinus.

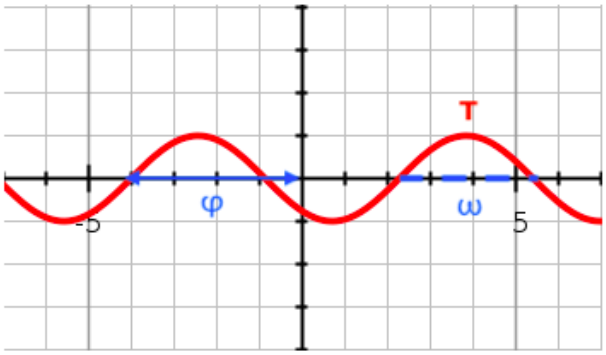

Figure 6 : Représentation 2D d'une caractéristique temporelle périodique.

On peut directement voir à la figure 6, provenant des résultats des auteurs de l'article « *Time2Vec : Learning a Vector Representation of Time* »[3], que la précision d'un réseau LSTM en combinaison avec différentes fonctions non linéaires du vecteur temps (*Time2vec*), que la fonction sinus est la plus performante comparativement à la fonction ReLU qui est la moins performante. La raison pour laquelle la fonction ReLU a des résultats aussi insatisfaisants est qu'une fonction ReLU n'est pas invariante en fonction de l'échelle du temps. Plus une fonction est invariante par rapport en fonction de l'échelle du temps, meilleures sont les performances.

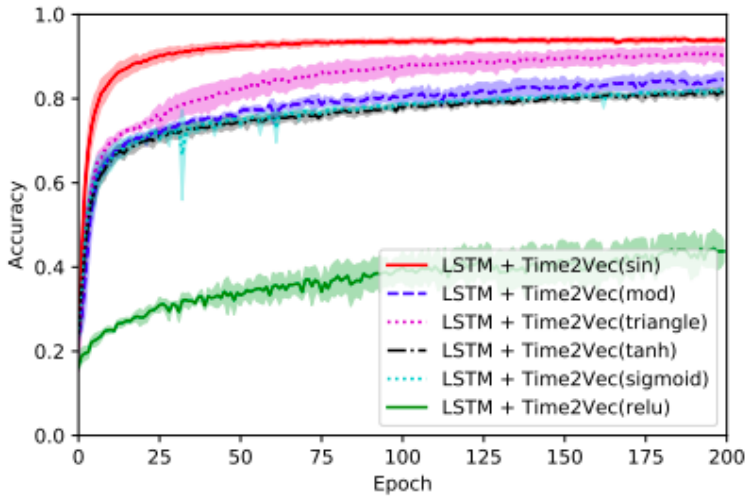

Figure 7 : Comparaison des performances des fonctions non linéaires.

[3] Kazemi, S.M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., & Brubaker, M.A. « *Time2Vec: Learning a Vector Representation of Time.* » 2019, URL : ArXiv, abs/1907.05321.

Transformers :

Maintenant que nous savons qu'il est important de fournir une notion de temps et comment implémenter un vecteur temporel, l'étape suivante est l'utilisation d'un modèle de type Transformers[4]. Ce modèle correspond à une architecture de réseau neuronal qui utilise un mécanisme d'auto-attention, permettant au modèle de se concentrer sur les parties pertinentes de la série temporelle pour améliorer les qualités de prédiction. Le mécanisme d'auto-attention se compose d'une couche d'attention à tête unique et d'une couche d'attention à têtes multiples. Le mécanisme d'auto-attention est capable de connecter toutes les étapes de la série temporelle les unes avec les autres en même temps, ce qui conduit à la création d'une compréhension de la dépendance à long terme. Enfin, tous ces processus sont parallélisés dans l'architecture Transformers, ce qui permet d'accélérer le processus d'apprentissage.

[4] VASWANI, Ashish, SHAZEER, Noam, PARMAR, Niki, et al. Attention is all you need. Advances in neural information processing systems, 2017, vol. 30.

Combination des données de RBC et des caractéristiques temporelles :

Après avoir implémenté les *Times Embeddings*, nous allons utiliser le vecteur temps en combinaison avec les caractéristiques de prix et de volume de RBC comme entrée pour notre Transformers. La couche *Time2Vector* reçoit les caractéristiques de prix et de volume de RBC en entrée et calcule les caractéristiques temporelles périodiques et non périodiques. Dans l'étape suivante du modèle, les caractéristiques temporelles calculées sont concaténées avec les caractéristiques de prix et de volume pour former une matrice, de la forme (32, 128, 7).

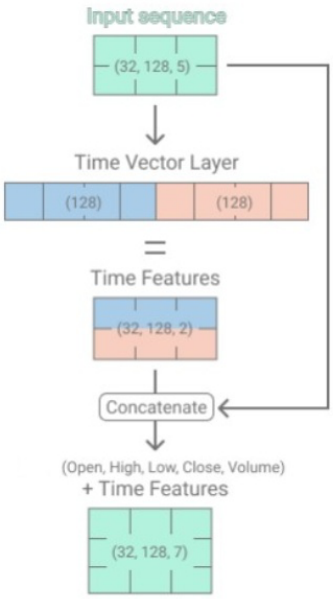

Figure 8 : Calcul des caractéristiques temporelles et concaténation avec les prix et le volume de RBC.

Single-Head Attention :

Les séries chronologiques de RBC et les caractéristiques temporelles que nous venons de calculer constituent l'entrée initiale de la première couche d'attention à tête unique. La couche d'attention à tête unique prend 3 entrées (*Query*, *Key*, *Value*) au total. Dans le cas de ce travail, chaque entrée *Query*, *Key* et *Value* est représentative des caractéristiques de prix, de volume et de temps de RBC. Chaque entrée *Query*, *Key* et *Value* reçoit une transformation linéaire distincte en passant par des couches denses individuelles (ici 96 cellules de sortie).

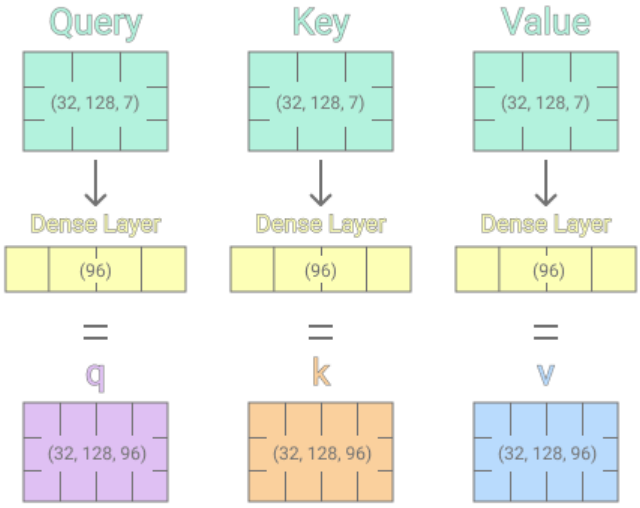

Figure 9 : Transformations linéaires des inputs (*Query*, *Key*, *Value*).

Après la transformation linéaire initiale, on calcule les poids d'attention. Les poids d'attention déterminent l'importance accordée aux étapes individuelles des séries temporelles lors de la prédiction d'un futur cours de bourse. Les poids d'attention sont calculés en prenant le produit scalaire des entrées *Query* et *Key* transformées linéairement, tandis que l'entrée *Key* transformée a été transposée pour rendre la multiplication du produit scalaire possible. Le produit scalaire est ensuite divisé par la dimension des couches denses précédentes (96), afin d'éviter l'explosion des gradients. Le produit scalaire divisé passe ensuite par la fonction softmax afin de produire un ensemble de poids dont la somme est égale à 1. Dans la dernière étape, on multiplie la matrice softmax (qui détermine la

focalisation de chaque étape temporelle), par la matrice v transformée, ce qui conclut le mécanisme d'attention à tête unique.

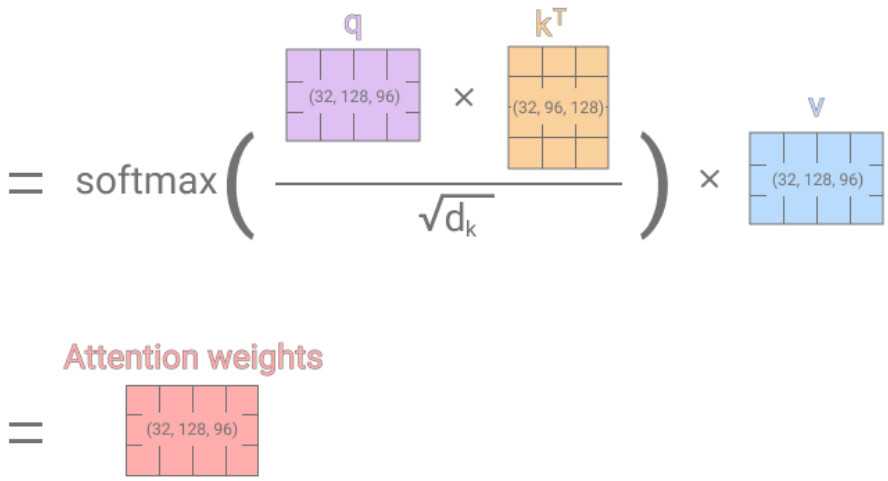

Figure 10 : Calcul des poids d'attention après transformation linéaire des inputs (*Query*, *Key*, *Value*).

Multi-Head Attention :

Pour encore améliorer le mécanisme d'auto-attention, les auteurs de l'article « *Attention Is All You Need* »[5] ont proposé la mise en œuvre d'une attention multi-têtes. La fonctionnalité d'une couche d'attention à têtes multiples consiste à concaténer les poids d'attention de n couches d'attention à tête unique, puis à appliquer une transformation non linéaire avec une couche dense. La figure 11 ci-dessous, montre la concaténation de 3 couches à tête unique.

Le fait de disposer de la sortie de n couches à tête unique permet d'encoder la transformation de plusieurs couches à tête unique indépendantes dans le modèle. Ainsi, le modèle est capable de se concentrer sur plusieurs étapes de séries temporelles à la fois.

[5] VASWANI, Ashish, SHAZEER, Noam, PARMAR, Niki, et al. « *Attention is all you need*. » Advances in neural information processing systems, 2017, vol. 30.

L'augmentation du nombre de têtes d'attention a un impact positif sur la capacité d'un modèle à capturer les dépendances sur de longs horizons temporels.

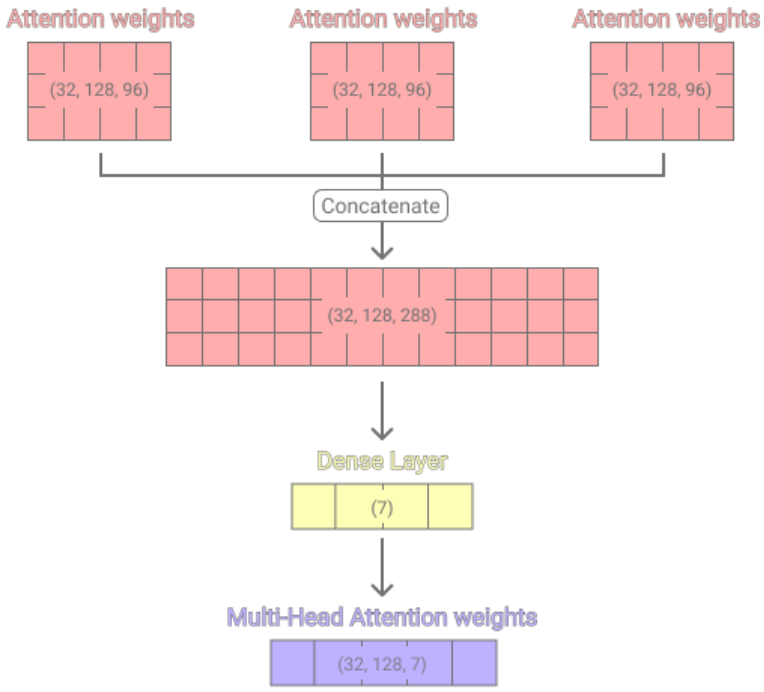

Figure 11 : Multi Attention Head Layer.

Transformer Encoder Layer :

Les mécanismes d'attention à une ou plusieurs têtes (auto-attention) sont maintenant regroupés dans une couche d'encodage transformatrice. Chaque couche d'encodage comprend une sous-couche d'auto-attention et une sous-couche d'anticipation. La sous-couche d'anticipation consiste en deux couches denses entre lesquelles se trouve une activation ReLU.

Chaque sous-couche est suivie d'une couche d'abandon, après l'abandon, une connexion résiduelle est formée en ajoutant l'entrée initiale de la requête aux deux sorties de la sous-couche. Pour conclure chaque sous-couche, une couche de normalisation est placée après l'ajout de la connexion résiduelle pour stabiliser et accélérer le processus de formation.

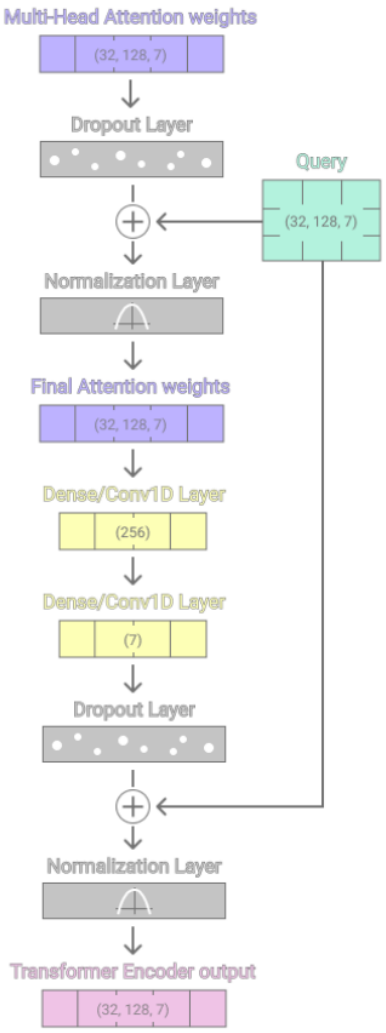

Figure 12 : Transformer encoder Layer.

Résultats :

Le processus d'apprentissage du modèle possède un total de 20 époques. Après l'apprentissage, nous pouvons voir sur la figure 13, que notre modèle Transformers ne prédit qu'une ligne plate qui est centrée entre les changements quotidiens du prix des actions. En utilisant uniquement l'historique des prix de l'action RBC, même un modèle Transformers est simplement capable de prédire la tendance linéaire de l'évolution des prix d'une action. On peut alors supposer que les données historiques de prix et de volume d'une action ne contiennent qu'une valeur explicative suffisante pour une prédiction de tendance linéaire.

Également il est à noter que les fonctions de pertes que nous avons utilisés ici sont la MSE, MAE et MAPE.

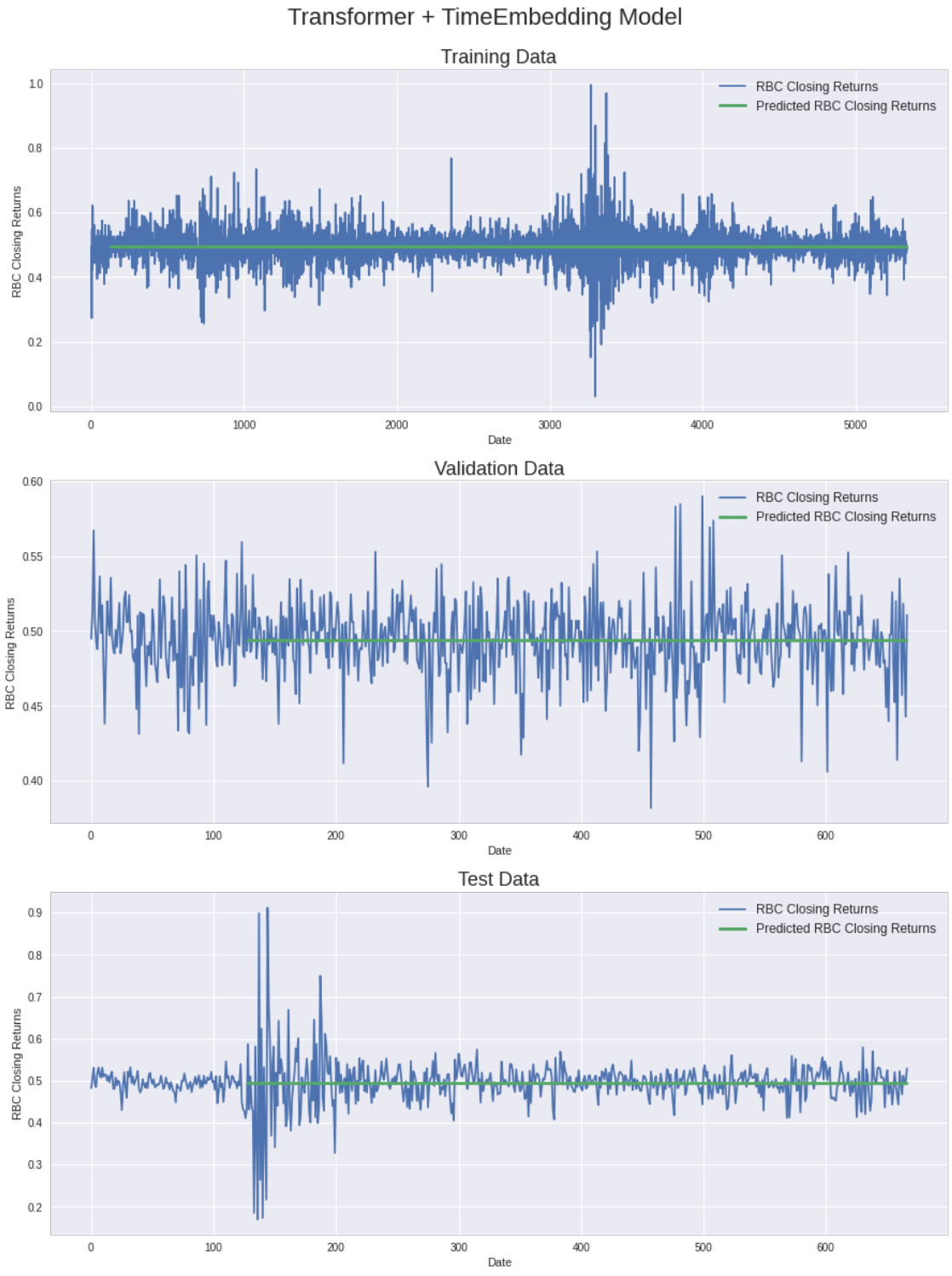

Figure 13 : Transformers + Time Embedding Model.

```
Evaluation metrics
Training Data - Loss: 0.0025, MAE: 0.0344, MAPE: 7.4334
Validation Data - Loss: 0.0007, MAE: 0.0189, MAPE: 3.8772
Test Data - Loss: 0.0032, MAE: 0.0341, MAPE: 7.6554
```

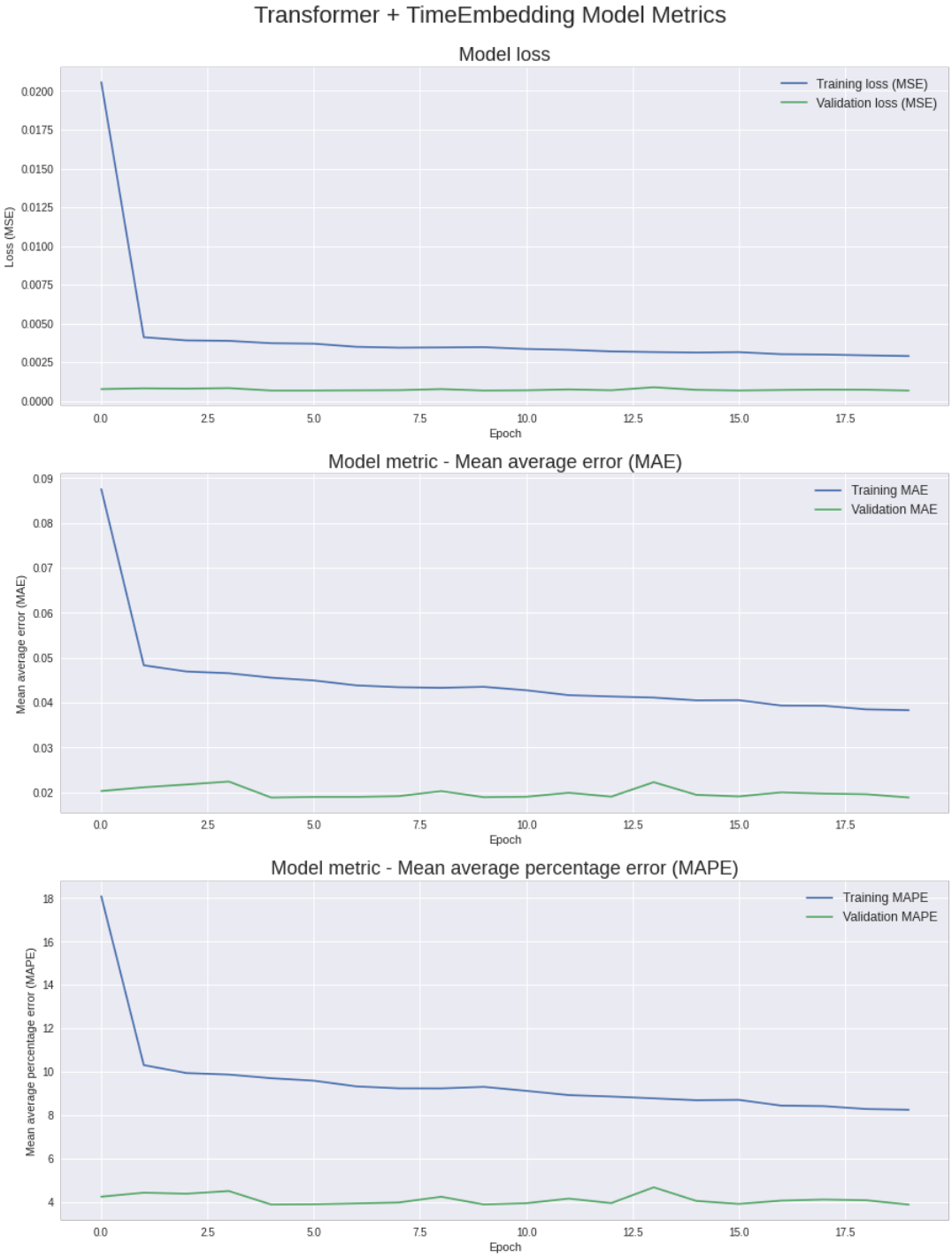

Figure 14 : Transformers + Time Embedding Model Metrics.

Comme indiqué ci-dessus, même les architectures de modèle les plus avancées ne sont pas en mesure d'extraire des prédictions non linéaires sur les actions à partir des prix et des volumes historiques des actions. Cependant, en appliquant un simple effet de lissage de la moyenne mobile sur les données (taille de la fenêtre = 10), le modèle est capable de fournir des prédictions nettement meilleures (ligne verte). Au lieu de prédire la tendance linéaire de l'action RBC, le modèle est capable de prédire également les hausses et les baisses, comme on peut l'observer sur la figure 15. Cependant, on peut noter que le modèle a un delta de prédiction important sur les jours avec un taux de changement quotidien extrême, ce qui indique que le modèle a des difficultés à prédire les valeurs aberrantes.

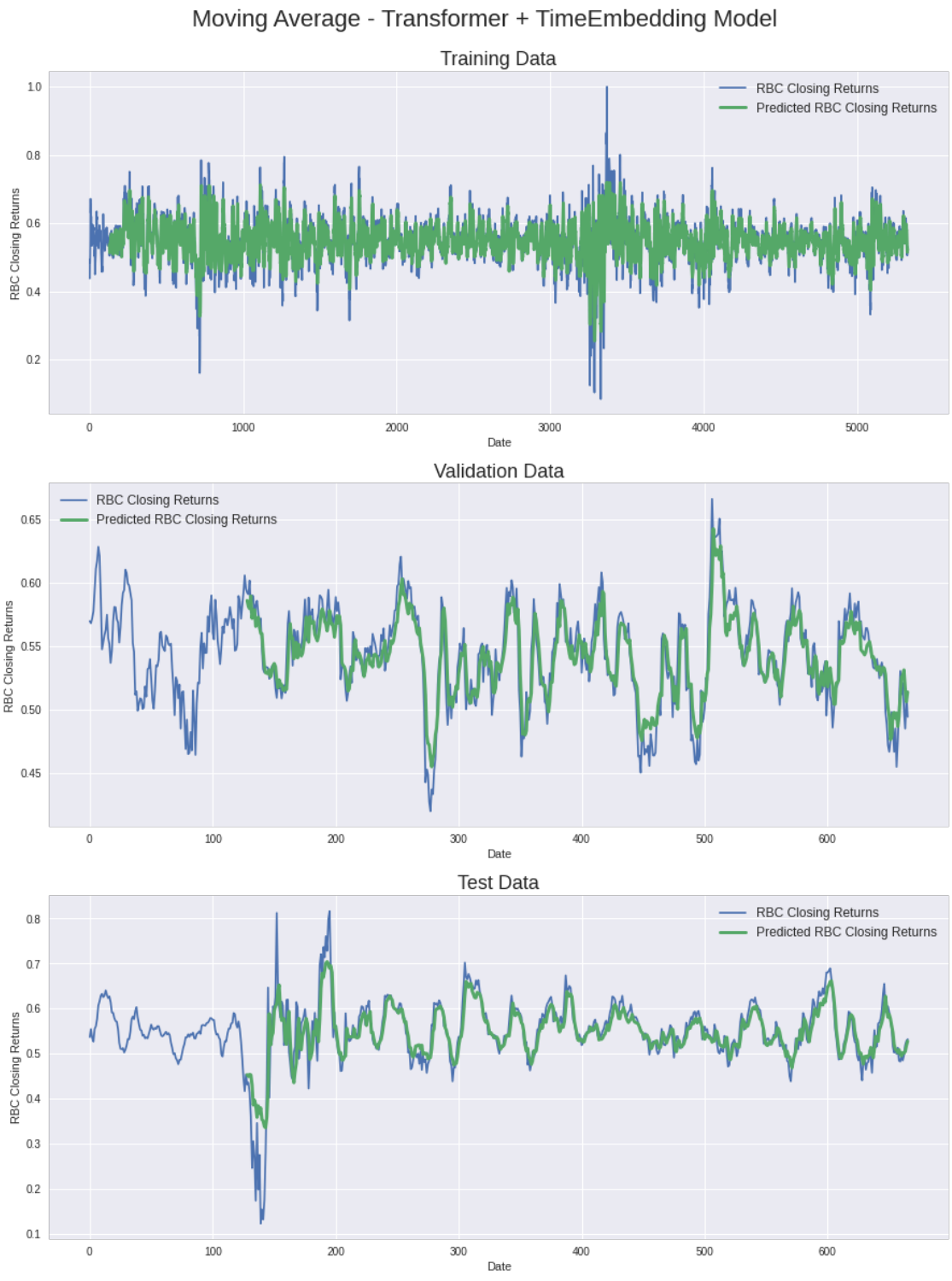

Figure 15 : Moving Average – Transformers + Time Embedding Model.

```
Evaluation metrics
Training Data - Loss: 0.0012, MAE: 0.0243, MAPE: 4.8025
Validation Data - Loss: 0.0003, MAE: 0.0135, MAPE: 2.5168
Test Data - Loss: 0.0016, MAE: 0.0242, MAPE: 5.6350
```

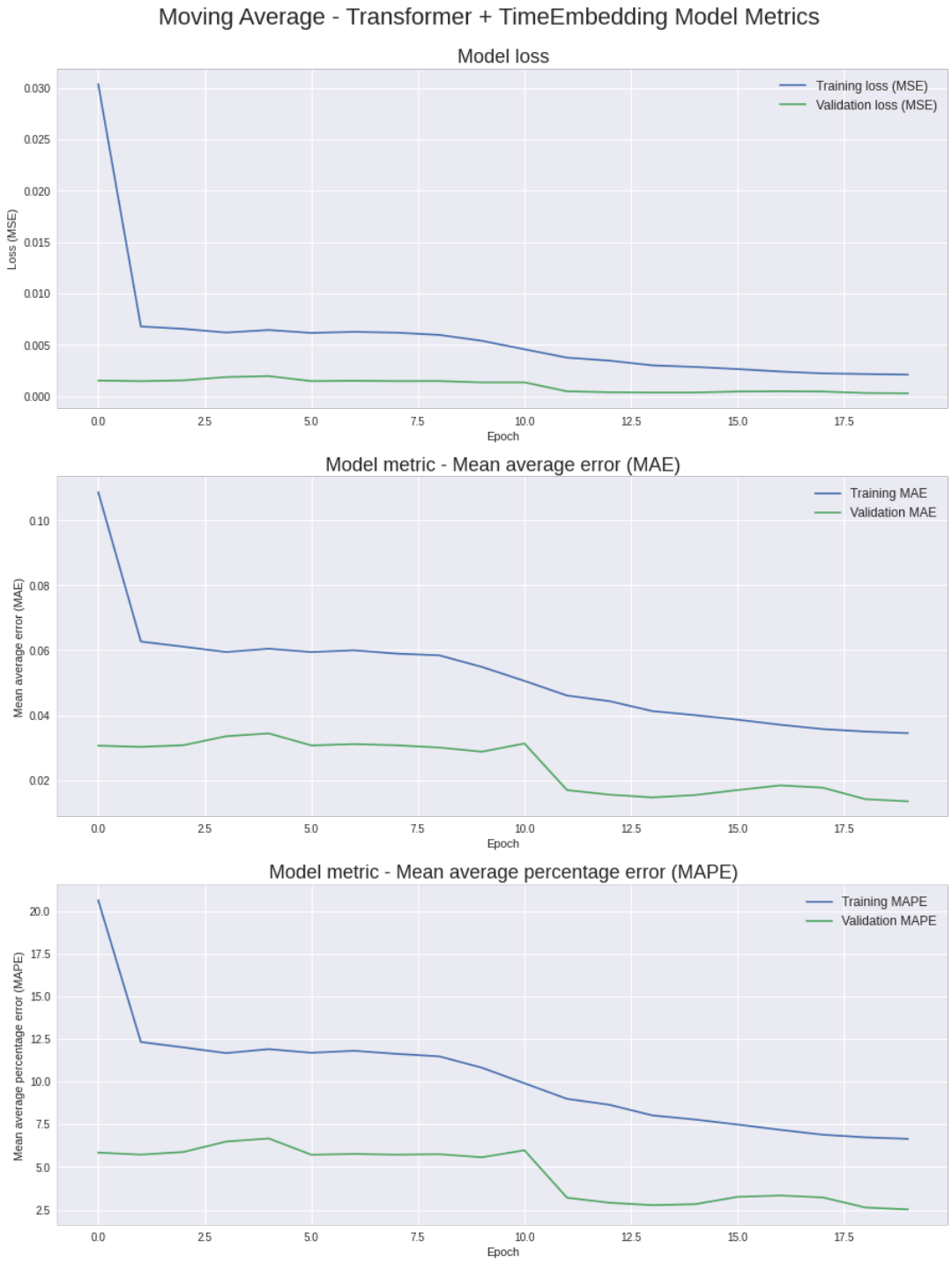

Figure 16 : Moving Average - Transformers + Time Embedding Model Metrics.

Références :

- Kazemi, S.M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., & Brubaker, M.A. « *Time2Vec: Learning a Vector Representation of Time.* » 2019, URL : ArXiv, abs/1907.05321.

- Tanya Goyal et Greg Durrett. « *Intégration d'expressions temporelles pour les modèles d'ordre temporel profond.* » 2019, Dans Actes de la 57e réunion annuelle de l'Association pour la linguistique computationnelle, pages 4400–4406, Florence, Italie. Association pour la linguistique computationnelle.

- VASWANI, Ashish, SHAZEER, Noam, PARMAR, Niki, et al. « *Attention is all you need*. » Advances in neural information processing systems, 2017, vol. 30.