



UNIVERSITY OF PIRAEUS

School of Information and Communication Technologies

Department of Informatics

Thesis

Thesis Title: Τίτλος Διατριβής:	KANB-Enhanced Neural Networks for Mobility Forecasting Νευρωνικά Δίκτυα Ενισχυμένα με KANB για Πρόβλεψη Κινητικότητας
Student's name-surname:	Kontoulis Markos
Father's name:	Vasileios
Student's ID No:	Π22074
Supervisor:	Yannis Theodoridis

September 2026/ Σεπτέμβριος
2026

Copyright ©

Copying, storing and distributing this work, in whole or in part, for commercial purposes is prohibited. Reproduction, storage and distribution for non-profit, educational or research purposes is permitted, provided the source is acknowledged and this notice is retained.

The views and conclusions contained in this document express the author exclusively and do not represent the official positions of the University of Piraeus.

As the author of this work, I declare that this work is not a product of plagiarism and does not contain material from unreferenced sources.

Abstract

This thesis examines whether trainable KANB activation functions improve short-term mobility forecasting when they replace the fixed activation in a Transformer feed-forward block. Ten Aegean and Brest grid-cell series are forecast at six horizons from 5 to 30 minutes, as well as the PEMS04 road-traffic benchmark. Hyperparameters were tuned sequentially on the validation set with KANB21 and five development series; the test split was not used during tuning.

The selected configuration for the maritime data uses Adam with a learning rate of 3×10^{-4} and $\epsilon = 10^{-5}$, batch size 256, 150 epochs, two Transformer blocks and four attention heads, without early stopping. The final comparison covers KANB17, KANB18, KANB21, ReLU, GELU and ParametricGELU over five random seeds, giving 300 successful runs. GELU has the lowest mean validation relative MAE (0.8037). On the test set, KANB17 has the lowest macro-average relative MAE (0.8498), while ReLU ranks first on five of the ten individual series. The KANB21–GELU test difference is only -0.0012 and is not statistically supported (exact $p = 0.912$). All nine KANB-versus-baseline confidence intervals include zero. The neural model beats persistence on eight series, but Brest 74768 and 75108 remain persistence-dominated. KANB is therefore a competitive low-parameter alternative in this setting, not a consistent improvement over standard activations. A validation-only follow-up with KANB17 tested batch size 512, ReduceLROnPlateau and a one-time learning-rate restart. None improved on the selected batch-256 constant-learning-rate protocol.

For the PEMS04 dataset, 71 features were used after being derived from observations, neighboring sensors, daily and weekly lags and calendar variables. The selection of the five sensors was done using training and validation data only. The test set wasn't used until the final comparison. In the final test, which contained 150 runs with early stopping and best-weight restoration, KANB18 had the lowest relative MAE (0.747688) against ReLU (0.749232) and GELU (0.750365). Against GELU, KANB18 performed better on all five sensors, 16 of 25 paired sensor-seed runs and five of the six forecast horizons. The aggregate improvement was 0.357%. The exact two-sided sensor-level test gave $p=0.0625$, and the confidence interval included zero.

Scientific Area: Artificial Intelligence, Machine Learning and Mobility Forecasting

Keywords: vessel traffic forecasting, AIS, Transformer, adaptive activation functions, KANB

Εκτεταμένη Περίληψη

Στη συγκεκριμένη εργασία εξετάζεται το αν εκπαιδευσιμες συναρτήσεις ενεργοποίησης της οικογένειας KANB μπορούν να βελτιώσουν την πρόβλεψη θαλάσσιας κινητικότητας. Το πρόβλημα είναι η πρόβλεψη του αριθμού πλοίων σε συγκεκριμένα γεωγραφικά κελιά για έξι χρονικούς ορίζοντες των 5 λεπτών, μέχρι τα 30 λεπτά. Οι δέκα χρονοσειρές που χρησιμοποιούνται είναι από την περιοχή του Αιγαίου και της Βρέστης, ενώ κάθε κελί αντιμετωπίζεται και χρησιμοποιείται για εκπαίδευση ως ξεχωριστό πρόβλημα. Στα δεδομένα περιλαμβάνονται ο τρέχοντας αριθμός πλοίων, οι χρονικές υστερήσεις, οι μεταβλητές γειτονικών κελιών και ημερολογιακά χαρακτηριστικά.

Εφόσον τα διαθέσιμα γνωρίσματα κάθε γειτονιάς δεν είναι κοινά και δεν έχουν το ίδιο σχήμα σε όλες τις περιοχές, η προεπεξεργασία των δεδομένων έγινε ανά κελί. Στο Αιγαίο υπήρχαν πολύ λίγα κενά, ενώ χρονικοί κάδοι που δεν υπήρχαν, αντιστοιχούσαν πρακτικά σε μηδενική κίνηση. Από την άλλη, στα δεδομένα της Βρέστης, οι μηδενικές τιμές υπήρχαν και αντιστοιχούσαν σε άγνωστες παρατηρήσεις. Έτσι έγινε μόνο περιορισμένη συμπλήρωση του μελλοντικού χρονικού ορίζοντα για κενά έως 15 λεπτά, με τα μεγαλύτερα διαστήματα να αφαιρούνται από την εκπαίδευση και την αξιολόγηση. Ο διαχωρισμός των δεδομένων έγινε με αναλογία 70/15/15 και ήταν χρονολογικός ώστε να μην εισάγεται πληροφορία από το μέλλον.

Το μοντέλο είναι ένας Transformer δύο επιπέδων. Κάθε αριθμητικό γνώρισμα μετατρέπεται σε διάνυσμα 64 διαστάσεων και αντιμετωπίζεται ως ξεχωριστό token. Επομένως, ο μηχανισμός προσοχής μαθαίνει σχέσεις ανάμεσα στα γνωρίσματα και όχι ανάμεσα σε διαδοχικά χρονικά βήματα. Η χρονική πληροφορία βρίσκεται στις μεταβλητές υστέρησης. Η συνάρτηση ενεργοποίησης που εξετάζεται τοποθετείται στο feed-forward τμήμα κάθε Transformer block, με ανεξάρτητες παραμέτρους ανά επίπεδο. Συγκρίνονται οι KANB17, KANB18 και KANB21 με τις ReLU, GELU και ParametricGELU. Όλες οι υπόλοιπες επιλογές του μοντέλου παραμένουν σταθερές.

Η πλήρης διαδικασία χωρίστηκε σε δύο στάδια. Αρχικά έγινε διαδοχική ρύθμιση υπερπαραμέτρων μόνο στο σύνολο επικύρωσης, με την KANB21 και πέντε χρονοσειρές ανάπτυξης. Κάθε φορά άλλαζε μία παράμετρος, ενώ το test set παρέμενε κλειστό. Επιλέχθηκαν learning rate 3×10^{-4} , Adam epsilon 10^{-5} , batch size 256, 150 epochs, depth 2 και 4 attention heads. Το early stopping δεν χρησιμοποιήθηκε και οι αρχικές τιμές των παραμέτρων KANB έμειναν αμετάβλητες.

Η τελική αξιολόγηση περιλαμβάνει 300 εκπαιδεύσεις: δέκα χρονοσειρές, έξι συναρτήσεις ενεργοποίησης και πέντε τυχαίους σπόρους. Η GELU έχει το χαμηλότερο μέσο validation relative MAE (0,8037). Στο test set, η KANB17 έχει το χαμηλότερο μέσο relative MAE (0,8498), ενώ η ReLU κατατάσσεται πρώτη σε πέντε από τις δέκα χρονοσειρές. Οι διαφορές είναι μικρές: για παράδειγμα, η διαφορά KANB21–GELU είναι -0,0012 και δεν είναι στατιστικά τεκμηριωμένη (exact p = 0,912).

Όλα τα διαστήματα εμπιστοσύνης των συγκρίσεων KANB με τις βασικές ενεργοποιήσεις περιλαμβάνουν το μηδέν. Το μοντέλο ξεπερνά την απλή πρόβλεψη επιμονής σε οκτώ χρονοσειρές, αλλά όχι στα Brest 74768 και 75108. Οι μαθημένες καμπύλες επιβεβαιώνουν ότι οι παράμετροι KANB ενημερώθηκαν σε κάθε Transformer block. Παρ' όλα αυτά, η προσαρμογή τους δεν έδωσε σταθερό πλεονέκτημα στο MAE. Το ασφαλές συμπέρασμα είναι ότι οι KANB είναι ανταγωνιστικές εναλλακτικές με μικρή επιβάρυνση σε παραμέτρους, όχι καθολικά καλύτερες συναρτήσεις ενεργοποίησης για το συγκεκριμένο πρόβλημα. Σε έναν πρόσθετο έλεγχο με την KANB17 δοκιμάστηκαν batch size 512, ReduceLROnPlateau και μία επαναφορά του learning rate. Καμία από αυτές τις αλλαγές δεν βελτίωσε τη μέση επίδοση στο validation set σε σχέση με το αρχικό πρωτόκολλο, δηλαδή batch size 256 και σταθερό learning rate.

Στην αξιολόγηση χρησιμοποιήθηκε επίσης το PEMS04, ένα σύνολο δεδομένων οδικής κυκλοφορίας με 307 αισθητήρες και παρατηρήσεις ανά 5 λεπτά. Κάθε χρονικό σημείο έχει 3 μεταβλητές: την κυκλοφοριακή ροή, πληρότητα και μέση ταχύτητα. Στόχος αυτού του πειράματος ήταν να αξιολογηθούν οι KANB σε ένα διαφορετικό πρόβλημα κινητικότητας, με μη γραμμική συμπεριφορά καθώς και χωρική διασύνδεση μεταξύ των αισθητήρων.

Καθώς η σύγκριση όλων των συνδυασμών μεταξύ των 307 αισθητήρων, με έξι ενεργοποιήσεις και 5 seeds θα χρειαζόταν 9210 runs, επιλέχθηκε ένα μικρό υποσύνολο. Η αρχική ομάδα υποψηφίων σχηματίστηκε μόνο από το training set, βάσει του απόλυτου και σχετικού

σφάλματος της πρόβλεψης persistence. Στο validation set κρατήθηκαν οι αισθητήρες στους οποίους το μη γραμμικό Histogram Gradient Boosting ξεπερνούσε το Ridge και το persistence στον ορίζοντα των 30 λεπτών. Με αυτή τη διαδικασία επιλέχθηκαν και πάγωσαν οι αισθητήρες 202, 289, 303, 2 και 106. Το test set δεν χρησιμοποιήθηκε κατά την επιλογή.

Κάθε δείγμα του PEMS04 περιλαμβάνει 71 γνωρίσματα: 36 πρόσφατες τιμές του αισθητήρα-στόχου, 24 πρόσφατες τιμές ροής από δύο γειτονικούς αισθητήρες, έξι ημερήσιες και εβδομαδιαίες υστερήσεις και πέντε ημερολογιακές μεταβλητές. Ο στόχος είναι η κυκλοφοριακή ροή στους ίδιους έξι χρονικούς ορίζοντες που χρησιμοποιήθηκαν στα ναυτιλιακά δεδομένα, από 5 έως 30 λεπτά.

Το τελικό πείραμα περιλάμβανε 150 runs, δηλαδή πέντε αισθητήρες, έξι συναρτήσεις ενεργοποίησης και πέντε τυχαίους σπόρους. Η KANB18 πέτυχε το χαμηλότερο μέσο test relative MAE, με τιμή 0,747688, έναντι 0,749232 της ReLU και 0,750365 της GELU. Η KANB18 ήταν καλύτερη από τη GELU και στους μέσους όρους των πέντε αισθητήρων, σε 16 από τα 25 paired runs και σε πέντε από τους έξι χρονικούς ορίζοντες. Η συνολική βελτίωση έναντι της GELU ήταν 0,357%.

Η διαφορά δεν έφτασε το συμβατικό επίπεδο στατιστικής σημαντικότητας, αφού ο ακριβής αμφίπλευρος έλεγχος έδωσε $p=0,0625$. Με μόνο πέντε ανεξάρτητους αισθητήρες, όμως, το πείραμα είχε περιορισμένη στατιστική ισχύ. Η κοινή κατεύθυνση του αποτελέσματος στους αισθητήρες και στους περισσότερους χρονικούς ορίζοντες αποτελεί θετική ένδειξη για την KANB18. Χρειάζεται αξιολόγηση σε μεγαλύτερο δείγμα αισθητήρων για να διαπιστωθεί αν το πλεονέκτημα διατηρείται και αποκτά ισχυρότερη στατιστική τεκμηρίωση.

Επιστημονική περιοχή: Τεχνητή Νοημοσύνη, Μηχανική Μάθηση και Πρόβλεψη Κινητικότητας
Λέξεις-Κλειδιά: πρόβλεψη θαλάσσιας κυκλοφορίας, AIS, Transformer, προσαρμοστικές συναρτήσεις ενεργοποίησης, KANB

Table of Contents

Abstract	ii
Εκτεταμένη Περίληψη.....	iii
Table of Contents.....	v
List of Figures	vii
List of Tables	viii
Introduction	1
Motivation and significance of the problem.....	1
Problem definition.....	1
Objectives and expected contribution.....	1
Thesis structure	1
1. Theoretical Background.....	2
1.1 Foundations of deep neural networks.....	2
1.2 Sequential and spatio-temporal models.....	2
1.3 From fixed to adaptive activation functions	3
1.4 Kolmogorov–Arnold Networks (KAN)	4
1.5 KANB: Parametric Bézier activation functions	4
2. Related Work	6
2.1 Maritime mobility forecasting with AIS data	6
2.2 Applications of adaptive activations in sequential models	6
2.3 Research gap	6
3. Methodology	8
3.1 Problem definition and input–output.....	8
3.2 Baseline architectures.....	8
3.3 Integrating KANB into the architectures.....	8
3.3.1 KANB variants	8
3.4 Datasets and preprocessing.....	9
3.5 Experimental design	11
4. Results and analysis	13
4.1 Main results on multi-horizon vessel-count forecasting	13
4.2 Controlled activation comparisons and baseline checks	15
4.3 Training dynamics and computational cost.....	16
4.4 Learned activation curves	17
4.5 Follow-up training experiments	19
Conclusions	22
Summary of findings.....	22
Limitations	23
Directions for future work	23
Glossary of Terms.....	25
List of Abbreviations and Acronyms.....	26

Bachelor Thesis

Markos Kontoulis

Bibliography.....

27

Appendices.....

28

Appendix A — KANB implementation excerpt.....

28

Appendix B — Detailed maritime activation-comparison results

28

Appendix C — Hyperparameter grids.....

29

List of Figures

Figure 3.1. Representative weeks from an Aegean and a Brest series.

Figure 3.2. Regional vessel-count distributions after preprocessing.

Figure 4.1. Average activation rank and number of series wins.

Figure 4.2. Test MAE by forecast horizon for Aegean grid 19285.

Figure 4.3. Skill relative to persistence for every activation and series.

Figure 4.4. Paired KANB effects with series-level bootstrap confidence intervals.

Figure 4.5. Seed variation and mean MAE on two high-headroom Aegean series.

Figure 4.6. Learned KANB curves in Transformer blocks 1 and 2.

List of Tables

- Table 3.1. Valid samples used for each grid-cell model.
- Table 3.2. Common experimental configuration.
- Table 3.3. Frozen PEMS04 experimental configuration.
- Table 4.1. Activation ranks across the ten series.
- Table 4.2. Best neural activation and persistence-relative skill by series.
- Table 4.3. Series-level paired relative-MAE differences.
- Table 4.4. Best neural and classical sanity-check model by series.
- Table 4.5. Training-time and validation-loss diagnostics across 50 runs per activation.
- Table 4.6. Validation-only KANB17 training follow-up.
- Table 4.7. Aggregate PEMS04 relative MAE across the frozen validation comparison, frozen final test, and post-test optimized validation-only sensitivity analysis.
- Table 4.8. Sensor-level PEMS04 final-test relative MAE for KANB18 and GELU.
- Table 4.9. PEMS04 final-test relative MAE by forecast horizon for KANB18 and GELU.
- Table B.1. Mean six-horizon test MAE by activation.
- Table C.1. Sequential tuning grid and selected final configuration.
- Table C.2. PEMS04 frozen protocol and post-test validation-only training sweep.

Introduction

Motivation and significance of the problem

Short-term mobility forecasts support decisions that have to be made before the next observation arrives: traffic monitoring, port coordination, route planning and early warning around congested sea areas. AIS feeds make this possible at a fine temporal resolution, but the resulting series are uneven. Busy cells show strong short-term fluctuations, while quiet cells may remain unchanged for long periods. A useful model must therefore learn genuine movement patterns without confusing missing observations with an absence of vessels. As far as road networks go, a forecasting model must be able to capture nonlinearity and mobility dynamics while remaining robust across locations, by supporting congestion monitoring and traffic management.

Most forecasting studies concentrate on the architecture used to represent space and time. The activation function inside that architecture is usually treated as a settled choice. ReLU and GELU are common defaults, although a fixed nonlinearity cannot adapt its shape to a particular dataset. KANB functions offer a controlled way to test whether that assumption matters: they add only a few trainable scalars to each layer, so a performance difference cannot be explained by a large increase in model capacity.

Problem definition

For every valid 5-minute observation at time t , the model receives the current count, lagged and neighboring-grid variables, and calendar information. It predicts six vessel counts, $y(t+1)$ through $y(t+6)$, corresponding to 5, 10, 15, 20, 25 and 30 minutes ahead. Counts are not normalized at the output. This keeps the evaluation metric interpretable as the average absolute error in vessels.

The central comparison holds the data, model width, attention mechanism, optimizer and training budget fixed. Only the activation in the Transformer feed-forward block changes. The main research question is: do KANB17, KANB18 or KANB21 reduce multi-horizon forecasting error relative to ReLU, GELU and a trainable ParametricGELU baseline? Three supporting questions concern whether any effect depends on the forecast horizon or series, what shapes the KANB functions learn, and whether the two Transformer blocks converge to different activation parameters.

Objectives and expected contribution

The first objective is to construct a leakage-safe forecasting pipeline for heterogeneous Aegean and Brest data. The second is to compare adaptive and fixed activation functions under a common experimental protocol and repeated random seeds. The third is to inspect the learned functions rather than treating KANB as a black box. The contribution is therefore empirical: it establishes where KANB is competitive, where simple persistence remains difficult to beat, and how large the activation effect is relative to variation across seeds and series.

Thesis structure

Chapter 1 introduces the neural-network, sequence-modeling and adaptive-activation concepts used in the study. Chapter 2 reviews vessel forecasting with AIS data and the smaller literature on trainable activations in time-series models. Chapter 3 describes the data pipeline, feature-tokenized Transformer and controlled experiment. Chapter 4 reports predictive accuracy, paired comparisons, training behavior and learned curves. The final chapter states the conclusions and limitations. Implementation details and full grid-level results are collected in the appendices.

1. Theoretical Background

1.1 Foundations of deep neural networks

To train a mobility forecasting model we rely on deep neural networks that can approximate non-linear mappings from high-dimensional inputs. To train these models we rely on gradient-based learning and backpropagation. LeCun et al. [1] showed how the chain rule can be used to compute gradients with very high accuracy, layer by layer. They did so by analyzing gradient-based learning for Convolutional Neural Networks (CNNs). In these networks, a convolutional layer takes an input tensor and produces a feature map as an output. It does that by applying the same set of kernels on every spatial location. After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model. This process gives a parameter count independent of the input resolution, and is one of the main reasons why CNNs scale so well on vision tasks [1].

The theoretical justification as to why using deep learning models as universal function approximators in forecasting problems — such as traffic and vessel mobility — is shown by the more recent mathematical work of Cheng et al. [2]. This paper showed that deep networks are so effective because under mild assumptions they can interpolate data, approximate smooth functions, and their dynamics under gradient descent can be analyzed in terms of controllability of the underlying system.

Creating deeper networks came with the degradation problem. The practitioners saw that by simply stacking more layers on a network, training and test accuracy were significantly reduced because gradients either vanished or exploded. He et al. [3] addressed this by introducing deep residual networks (ResNets), where a residual block computes:

$$y_{l+1} = F(x_l; W_l) + x_l \quad (1.1)$$

i.e., the layer learns a residual correction on top of the identity. These connections work like shortcuts that stabilize gradient flow and allow the training of very deep models, which is very important when capturing patterns in complex mobility data (e.g., short-term local fluctuations vs. long-term global trends).

These architectures establish how depth, nonlinearity and stable gradient flow combine to make neural networks effective function approximators. They do not, however, address a defining feature of mobility data: that the future depends not on a single snapshot but on a sequence of past states.

1.2 Sequential and spatio-temporal models

Since the future trajectory in mobility is not dependent on a snapshot but a sequence of past states, that makes mobility forecasting inherently a temporal problem. Elman's Simple Recurrent Network (SRN) [4] formalized this idea. At each time step, the network updates a hidden state:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1.2)$$

and outputs a prediction based on it. This recurrent structure allows the model to “remember” previous inputs and discover structure in time.

Due to the degradation problem, however, standard RNNs struggle with long-term dependencies. Long Short-Term Memory (LSTM) networks, proposed by Hochreiter and Schmidhuber [5], introduced gating mechanisms — input, forget, and output gates — that could control what happened with the information. In a simplified form, the cell update is

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (1.3)$$

where the terms are gate activations. This architecture can preserve information over long horizons and is therefore widely used for trajectory and traffic forecasting.

Mobility data, however, are not only structured in time, but in space too, since road networks, shipping lanes or sensor networks define graphs rather than Euclidean grids. So there is a need for another way to address them, other than RNNs and LSTMs. Jiang and Luo [6] survey Graph Neural Networks (GNNs) for traffic forecasting and show that, for many problems, standard CNN/RNN architectures are inadequate because they assume regular grids and local neighborhoods defined by convolutions.

GNNs model graphs where nodes represent road segments or sensors, and edges usually represent connectivity or interaction. Gilmer et al. [7] introduced a message-passing framework in which each layer updates node embeddings by aggregating messages from neighbors:

$$h_v^{(k+1)} = \phi(h_v^{(k)}, \oplus_{u \in \mathcal{N}(v)} \psi(h_v^{(k)}, h_u^{(k)}, e_{uv})) \quad (1.4)$$

where $\oplus$ is an aggregation operator (e.g., sum or mean). In the context of mobility, the model can now capture how congestion propagates outwards, towards the connected neighbors, integrating spatial topology directly into the learning process [6].

With the creation of transformers [8], recurrence was replaced with self-attention. For queries Q , keys K , and values V , a self-attention layer computes:

$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (1.5)$$

so each time step attends to all others and attention weights are determined by how similar the keys are in the key space. This process is done over several subspaces and the results are later concatenated, capturing different interaction patterns.

Lee [9] uses transformer-based models at urban, low-speed intersections in an attempt to predict vehicle trajectories, and shows that they outperform classic RNN/LSTM models, with the difference being more noticeable as the driving scenarios and their complexity got higher. The problem with that, however, is that transformer models have quadratic time complexity. Given a token matrix X , self-attention computes:

$$Q = XW_Q, K = XW_K, V = XW_V, \text{Att}(X) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1.6)$$

Self-attention thus offers a flexible way to model dependencies across a sequence without the sequential bottleneck of recurrence, which is why the Transformer is adopted as the base architecture in this thesis. All of these models, however — recurrent, graph-based and attention-based alike — share one design element that is almost always fixed in advance: the nonlinear activation function.

1.3 From fixed to adaptive activation functions

Most of the architectures mentioned often rely on nonlinear activation functions such as ReLU, Tanh or Sigmoid. The choice of activation function for a model can affect accuracy, convergence speed and misclassification confidence in CNN text classification. Emanuel et al. [10] show that even when the architecture and data remain fixed, just the choice of activation function can significantly impact the model's performance, meaning that defaulting to the "standard" can sometimes be suboptimal for certain tasks.

This observation has created a huge body of work on the design of activation functions. Apicella et al. [11] separate fixed activations from trainable ones, whose shape is adjusted during training instead of remaining fixed or being chosen beforehand. Dubey et al. [12] survey this comprehensively and benchmark different activation families, reporting that no single fixed activation dominates across tasks. Evidently, the best choice depends on the nature of the problem. A well-known step toward trainable nonlinearities is Swish [13], defined as

$f(x) = x \cdot \sigma(\beta x)$, where β is a learnable parameter that lets the network interpolate between a near-linear and a strongly gated response. Swish keeps the functional form fixed, while allowing for a single scalar of its shape to become learnable, something which shows somewhat of an improvement over ReLU on deep models.

The shape of the activations also affects and interacts with the rest of the network. Joudaki et al. [14] study how activations and normalization affect the geometry of representations at

initialization. They also show how certain combinations preserve distances better than others, a property that matters for stable, trainable deep networks. Generative Adversarial Networks (GANs) [15], as well as diffusion models [16] further demonstrate how sensitive deep learning architectures are to their nonlinear design choices.

What these approaches have in common is that the nonlinearity itself stays fixed or is made adjustable through a single scalar. A more recent line of research, however, aims to answer the question of whether the entire activation function itself can be made learnable and, so that the network can discover the shape of its own nonlinearity that best fits the problem and the data. This is the idea behind Kolmogorov-Arnold Networks (KAN) and the KANB family of activation functions, described in the following sections.

1.4 Kolmogorov–Arnold Networks (KAN)

The works above still treat activation functions as fixed. A more recent line of research aims to make the non-linearity itself learnable. Liu et al. [17] come up with the Kolmogorov-Arnold Networks (KAN), based on the Kolmogorov-Arnold representation theorem. Unlike traditional Multilayer Perceptrons (MLPs), that have fixed activation functions placed on each node (“neurons”), KANs have learnable activation functions on their edges. These functions are univariable and parametrized, often by B-splines. This way the network can adapt its non-linear transformations to the data. One can also understand how a particular input dimension is warped before aggregation by observing the learned spline on each edge.

As KAN shows, replacing fixed activations with trainable and flexible spline functions can improve accuracy and interpretability. Its formulation places these functions on the network’s edges and parameterizes them with splines; the KANB family, described next, retains the principle of a learnable nonlinearity but realizes it differently — with a Bézier-based parametric form applied directly at each neuron.

1.5 KANB: Parametric Bézier activation functions

Kanakoudis et al. [18] build on KAN by introducing a new family of activation functions for deep neural networks. Instead of splines on edges, however, KANB applies a Bézier-based parametric polynomial non-linearity directly on each neuron. First, for every pre-activation y , a generic KANB function maps y through a sigmoid:

$$t = \sigma(y) = \frac{1}{1+e^{-y}} \in (0,1), \quad \frac{dt}{dy} = t(1-t) \quad (1.7)$$

and then evaluates a cubic Bézier polynomial with trainable control points:

$$KANB(y) = 3(1 - \sigma(y))^2 \sigma(y) Y_{p1} + 3(1 - \sigma(y)) \sigma(y)^2 Y_{p2} + g(y) \sigma(y)^3 + f(y) \quad (1.8)$$

here Y_{p1} , Y_{p2} are trainable control points, and f , g specify different KANB variants. It is ensured that the curve is evaluated on the interval by using $\sigma(y)$ as the Bézier parameter, while the control points determine the shape of the non-linearity. The control points are ordinary trainable scalars updated via backpropagation along with the weights and the biases of the network. In the implementation used here, they were confined to the range $[-20,20]$ and initialized to small values. This way, the network learns how to combine its own features through its weights and how to bend the activation curve.

A point that matters while interpreting the results is that the trainable parameters are defined per layer, not per neuron or edge. A KANB layer only adds a small, fixed number of scalars, regardless of width. Meaning, that the parameter overhead compared to a fixed activation function is negligible, so the performance difference cannot be attributed to a larger parameter count.

$$KANB1(y) = 3(1 - \sigma(y))^2 \sigma(y) Y_{p1} + 3(1 - \sigma(y)) \sigma(y)^2 Y_{p2} + \sigma(y)^3. \quad (1.9)$$

$$\text{KANB2}(y) = 3(1 - \sigma(y))^2 \sigma(y) Y_{P1} + 3(1 - \sigma(y)) \sigma(y)^2 Y_{P2} + y \sigma(y)^3. \quad (1.10)$$

$$\text{KANB3}(y) = \text{KANB1}(y) + y. \quad (1.11)$$

$$\text{KANB4}(y) = \text{KANB1}(y) + \ln(1 + e^y). \quad (1.12)$$

In training, both Y_{P1} and Y_{P2} are updated via backpropagation, making the network learn the feature weights and the shape of the activation curve. For example, KANB1 corresponds to the pure Bézier non-linearity (no extra term), while KANB3 adds a residual identity term $f(y) = y$, making the activation non-saturating and improving gradient flow (its derivative is the derivative of KANB1 plus 1). Most KANB derivatives remain bounded with appropriate parameter constraints, reducing the risk of exploding gradients during training.

KANB is then tested on computer-vision benchmarks such as MNIST, CIFAR-10, CIFAR-100, and Cats vs Dogs, as well as on the Aegean–Cyclades vessel-traffic forecasting dataset. On the image classification task, on the CIFAR-100 dataset, KANB3 with trainable parameters achieves an accuracy of 80%, while ReLU, Tanh and Sigmoid only get 50–51%. This shows that Bézier functions can sometimes significantly outperform fixed ones on complex datasets. On the Aegean–Cyclades dataset, the KANB3 model manages to achieve a reduction in Relative Average Error by approximately 14% compared to a ReLU baseline across predict-ahead times of thirty minutes [18]. That demonstrates that adaptive polynomial activations capture the nonlinear dynamics underlying real-world maritime mobility data better. KANB therefore combines several desirable properties for mobility forecasting such as adaptivity, stability in optimization as well as compatibility with other preexisting architectures — meaning it can replace Tanh, Sigmoid or ReLU in LSTMs and GNNs.

This thesis doesn't use the four base variants directly, but three members of the extended KANB family – KANB17, KANB18 and KANB21 – selected on the basis of prior benchmark results. As implemented, their definitions are

$$\text{KANB}_{17}(y) = a[2\sigma(y)(1 - \sigma(y)) Y_{P1} + \sigma(y)^2] \quad (1.13)$$

$$\text{KANB}_{18}(y) = 3(1 - \sigma(y))^2 \sigma(y) Y_{P1} + 3(1 - \sigma(y)) \sigma(y)^2 Y_{P2} + \sigma(y)^3 + \max(0, y) \quad (1.14)$$

$$\text{KANB}_{21}(y) = \max(a[2\sigma(y)(1 - \sigma(y)) Y_{P1} + \sigma(y)^2], b \cdot y) \quad (1.15)$$

where a and b are additional trainable scalars. KANB18 follows the full cubic form of KANB1 with an added rectified-linear term, keeping it non-saturating. KANB17 and KANB21 instead build on a scaled quadratic Bézier term; KANB21 takes the maximum of that term and a linear response, giving it a piecewise, ReLU-like character.

2. Related Work

2.1 Maritime mobility forecasting with AIS data

AIS data record the position of vessels, their speed, their direction and related navigation information at irregular intervals. Studies on forecasting, have to first, convert those messages into representations from which our model can learn. One option is trajectory prediction, where the target variable is a sequence of coordinates where the vessel might find itself in the future. Another option is traffic-flow forecasting, where positions are placed into special cells and the model predicts flows or counts. The present work uses the second formulation, since the datasets that were used, already describe vessel counts in 5-minute grid-cell bins.

The spatial and temporal structure of the task is reflected by the literature. Evmides et al. [19], discuss the operational meaning of prediction horizons, while Liu et al. [20] use a spatio-temporal multi-graph Transformer for trajectory prediction, using interactions among several vessels for their model. Mandalis et al. [21] apply a Transformer-based method to vessel traffic-flow forecasting, which is especially close to the count-prediction setting considered here. Sometimes, as earlier work shows [22], maritime streams require careful handling of time, uncertainty and observations that might be incomplete. Broader reviews by Bao and Wang [23] and the mobility-analytics volume edited by Vouros et al. [24] place these models within the AIS processing pipeline.

A problem that keeps arising is that a strong baseline can be extremely simple. At a short horizon, we can get a good enough estimate of the next count by looking at the current count. Performance should therefore be analyzed and looked at against persistence, not only by looking at the absolute MAE. That baseline can also expose certain differences among cells. For example, a busy and variable cell offers the model room for improvement, while a sparse or stable series can be dominated by a simple no-change prediction.

2.2 Applications of adaptive activations in sequential models

Trainable activation functions have received far more attention in image classification than in mobility forecasting. Existing surveys describe approaches ranging from conventional activation functions with a single learned slope to spline- and polynomial-based functions with several parameters [11, 12]. No single design performs best across every architecture and dataset. Its effectiveness depends on factors such as the distribution of pre-activation values, network depth, normalization, and the optimization procedure.

Research on Kolmogorov–Arnold Networks (KANs) takes this idea further by learning univariate transformations instead of selecting them beforehand. Xu et al. [25] apply KANs to time-series data and argue that these flexible functions can offer both predictive accuracy and some interpretability. KANB is motivated by a similar idea, but it can be incorporated into an existing architecture more easily. It replaces a node activation while adding only a few trainable scalars at the layer level [18]. The Transformer’s feed-forward network can therefore be modified without changing its attention mechanism.

Still, making an activation function trainable does not guarantee better results. The activation introduces an additional optimization problem, and its flexibility may be redundant if the surrounding layers can already rescale their internal representations. ParametricGELU is included to examine this possibility. It helps distinguish the broader effect of learning activation parameters from any benefit associated specifically with the Bézier functions used in KANB.

2.3 Research gap

Previous maritime forecasting studies have compared sequence-based and graph-based architectures, but they have not treated the activation function as an isolated experimental factor. KANB, meanwhile, has produced strong results on image benchmarks and on an earlier forecasting task in the Aegean [18]. Evidence from multiple maritime series, forecast horizons,

and random seeds remains limited. It is therefore uncertain whether these improvements carry over to a feature-tokenised Transformer or depend on particular datasets and model designs.

This thesis investigates that question through repeated, paired comparisons across ten series drawn from two regions. Each activation is evaluated with identical data splits and random seeds. Cells with little room to outperform the persistence baseline are retained, since an inability to beat persistence is itself relevant to the forecasting task. The learned activation curves are also examined separately for the two Transformer blocks. This makes it possible to determine whether KANB uses its layer-specific parameters in practice.

3. Methodology

3.1 Problem definition and input-output

Let x_t denote the feature vector at the end of the 5-minute bin t that is available. Our target will then be the vector $y_t = [c_{(t+1)}, \dots, c_{(t+6)}]$. Here, c is the vessel count. A row can be included in the dataset only if it has no missing input values at the time, if the current bin is an observed bin, all the targets are observed, and both the targets and the input are in the same chronological split. This requirement, that all horizons are valid, gives each run a common test set for each of its outputs.

The output of the model is six real values, and since negative predictions have no meaning to vessel count, they are clipped to zero before MAE is calculated. The main metric used is $MAE_h = \text{mean}(|c_{(t+h)} - \hat{c}_{(t+h)}|)$ for each of the horizons h .

3.2 Baseline architectures

The baseline of the neural network is a Transformer encoder, comprised of two blocks. The processed rows are not raw sequences but flat feature vectors. A high-dimensional vector is mapped down to a scalar using a learned scalar projection. This turns each feature into a 64-dimensional token. Then, the variable that is represented by each of the tokens is identified by a learned feature-position embedding. Four-head, non-causal self-attention then captures relationships among the feature tokens. Inside every transformer block exists a hidden dimension of 128, residual connections and dropout set to 0.1. In-between the two blocks and the linear output head where the six forecasts are produced, global average pooling combines the feature-token representations.

This architecture can be described as feature-tokenized. Attention doesn't move over time steps, meaning that each token represents a separate feature in the input vector x_t . For each grid-cell of the series a model is trained separately. This way no parameters are shared between Aegean, Brest or neighboring cells.

The fixed, conventional comparison activations used are ReLU, GELU - commonly used in Transformer feed-forward layers, and ParametricGELU which adds trainable coefficients to GELU and controls for the possibility that any learned activation function is beneficial. Outside the neural comparison, ordinary persistence, lag-4 seasonal persistence, Ridge regression and histogram gradient boosting provide forecasting sanity checks.

3.3 Integrating KANB into the architectures

The activation under test is inserted between the two dense transformations of each Transformer block. Blocks 1 and 2 receive separate activation layers. Parameters are shared across the 128 hidden units within a block, but they are not shared between blocks. The parameter overhead is therefore constant with respect to model width: KANB17 adds two scalars per block, KANB18 adds two, and KANB21 adds three.

The KANB parameters are trained jointly with the rest of the network through backpropagation and are constrained to the interval $[-20, 20]$. The initialization is fixed for every run of a given variant: KANB17 starts with $YP1 = -0.5$ and $a = 3$; KANB18 with $YP1 = -0.1$ and $YP2 = -0.2$; and KANB21 with $YP1 = -0.5$, $a = 3$ and $b = 1$. As a result, the random seed changes the initial network weights and dropout patterns, while leaving the starting activation shape unchanged.

3.3.1 KANB variants

KANB17 first maps the pre-activation through a sigmoid and then applies a scaled quadratic Bézier term. This way, its positive side rises but then approaches a finite ceiling. KANB18 retains a non-saturating positive branch by adding a cubic Bézier correction to ReLU. KANB21 can behave like a smooth gate around zero while also remaining almost linear for sufficiently large positive inputs, by taking the maximum of a quadratic branch and a linear branch.

3.4 Datasets and preprocessing

The data used for the study include four Aegean grid cells (19285, 19653, 19837 and 20017) and six Brest cells (74429, 74767, 74768, 74769, 75108 and 75447). Each source file is resampled onto a 5-minute `DatetimeIndex`. Between regions and datasets, target names and calendar fields were different, so they were mapped explicitly instead of relying on the raw schema. Columns that neighbor each other contain absolute grid identifiers. The processing of the data is performed separately for each grid to prevent the structural missing values that would result from combining schemas that are not related and incompatible. Missing bins required treatment specific to the region and the nature of the dataset. In the Aegean files, there were no columns with the value of zero, while only seven timestamps were absent, which were filled with zero. This indicated that zero-traffic columns had already been removed from the dataset. On the other hand, the Brest files contained explicit zero counts in around 30% of rows, meaning that missing timestamps didn't mean there was also zero traffic. Gaps of at most three bins were forward-filled while longer sequences of gaps were simply excluded by the validity mask. Four Brest cells with more than 30% missing bins were removed before modeling.

Splits are chronological. The first 70% of the dataset is used for training, the next 15% for validation and the remaining 15% for testing. If a split boundary is crossed the sample is removed, since each sample must belong entirely to one dataset split. Standardization parameters are fitted only on complete training inputs and then applied to scale the validation and test data, avoiding data leakage. Targets were kept in raw counts.

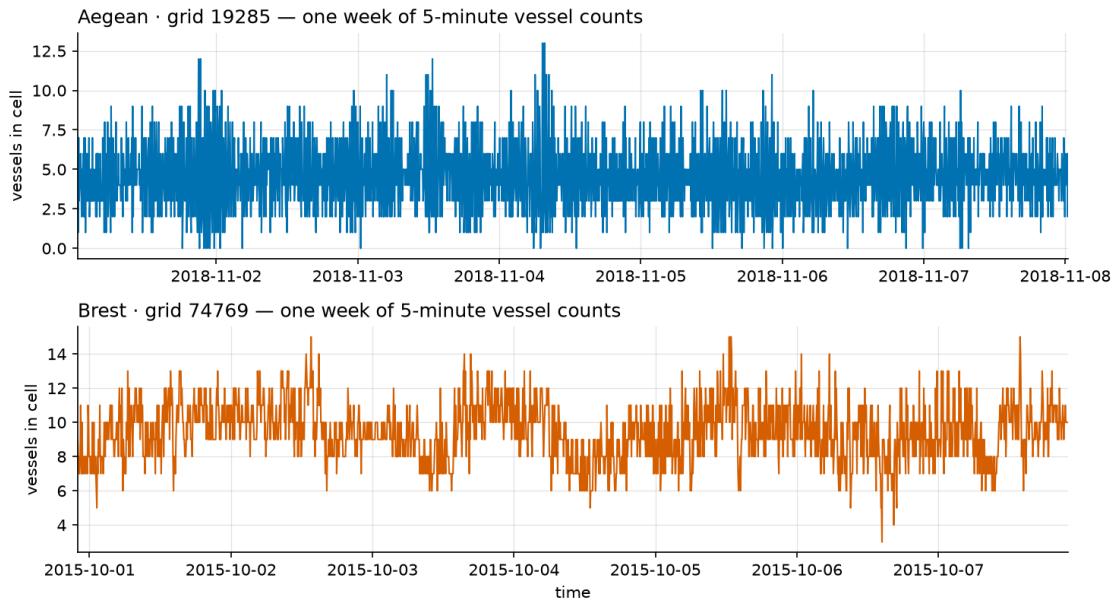


Figure 3.1. Representative weeks from an Aegean and a Brest series.

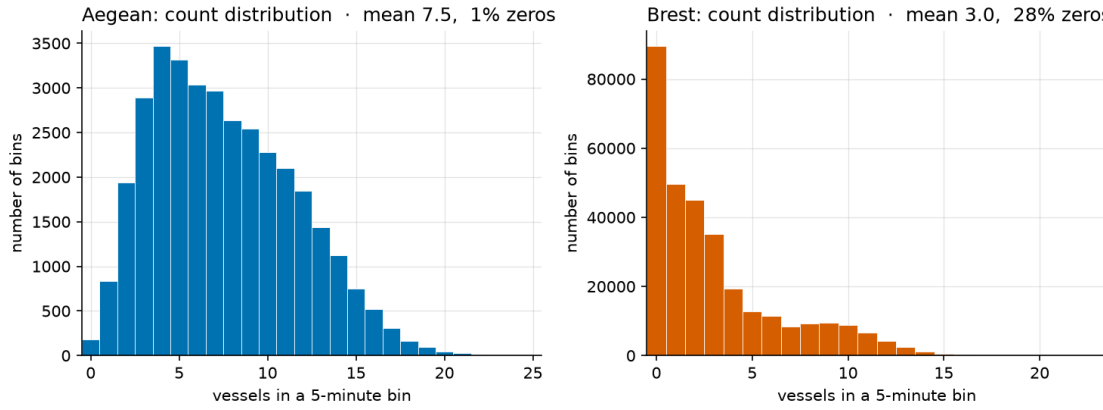


Figure 3.2. Regional vessel-count distributions after preprocessing.

Table 3.1. Valid samples used for each grid-cell model.

Region	Grid	Features	Train	Validation	Test
Aegean	19285	28	5,995	1,282	1,290
Aegean	19653	28	6,037	1,289	1,290
Aegean	19837	28	6,037	1,289	1,289
Aegean	20017	28	6,037	1,289	1,289
Brest	74429	59	36,599	7,858	7,900
Brest	74767	59	36,471	7,858	7,900
Brest	74768	59	36,599	7,858	7,900
Brest	74769	59	36,599	7,858	7,900
Brest	75108	59	36,599	7,858	7,900
Brest	75447	59	36,599	7,858	7,530

3.4.1 PEMS04 road-traffic benchmark

PEMS04 is the processed PeMSD4 road-traffic benchmark derived from the California Department of Transportation’s Performance Measurement System. PeMS collects loop-detector measurements across California freeways [26]. The version used here covers 1 January to 28 February 2018 and contains 16,992 five-minute observations for 307 sensors. Each observation records total flow, average occupancy and average speed, following the 307-sensor preprocessing described by Guo et al. [27].

The prediction target is traffic flow at the same six horizons used for the maritime data, from 5 to 30 minutes ahead. Each sample contains 71 inputs. The target sensor contributes flow, occupancy and speed at the forecast origin and the previous eleven timestamps, giving 36 values. Two graph neighbours contribute twelve recent flow readings each, for another 24 values. Daily and weekly lags add six measurements, while five calendar variables encode time of day, day of week and weekends. Neighbour readings enter as ordinary features: the model is still trained separately for each target sensor and does not use graph convolution.

A full six-activation, five-seed comparison over all 307 sensors would require 9,210 model fits for a single protocol, so the sensor subset was fixed before test evaluation. A training-only

persistence screen produced the candidate set. Validation data were then used to retain sensors on which Histogram Gradient Boosting outperformed Ridge regression and persistence at the 30-minute horizon. This procedure selected sensors 202, 289, 303, 2 and 106. Test performance played no part in the choice.

Samples were split chronologically into 70% training, 15% validation and 15% test partitions. Samples that crossed a split boundary were removed, and standardization was fitted on the training partition only. This left 9,872 training, 2,543 validation and 2,543 test samples per sensor.

3.5 Experimental design

Before the final comparison of the activation function, hyperparameter tuning was performed on KANB21 and five development series (4 cells Aegean and Brest 74769). As a metric for tuning the parameters, only validation relative MAE was used. While performing the tests, the factors were changed one at a time in the following order: learning rate, Adam epsilon, batch size, epoch budget, early stopping and finally depth and attention heads. The KANB initialization constants that can be found in Section 3.3 were left unchanged.

The configuration of choice was Adam with a learning rate of 3×10^{-4} and $\varepsilon = 10^{-5}$, a batch size of 256, 150 epochs, depth 2 and 4 attention heads. No early stopping was used, since the fixed 150-epoch protocol gave the better result in the experimental comparison. In the final experiment, 300 successful runs were completed (10 series $\times$ 6 activations $\times$ 5 seeds). On every run the architecture as well as the training budget remained unchanged, with the only variable being the activation layer. After that, a validation only follow-up comparison, used KANB17 on the same 5 development series and seeds, comparing this protocol to a different one which had a batch size of 512 and used ReduceLROnPlateau instead of a constant learning rate. It was also compared against a different protocol which used ReduceLROnPlateau and then followed by one restart to the initial learning rate. These experiments, however, did not lead to any changes in the final result and the comparisons of the activation functions.

MAE is reported in vessel counts within each series. For cross-cell comparisons, however, the main measure of performance is $relative\ MAE = \frac{MAE_{model}}{MAE_{baseline}}$ and skill defined as one minus that ratio. If the value is less than one, then there's an improvement of the model on the no-change forecast. Contrasts in activations are paired and compared by series and seed. Differences are averaged within each series before a 20,000-sample bootstrap confidence interval and a two-sided exact sign-flip test are calculated, so horizons and seeds from the same series are not treated as independent datasets.

Table 3.2. Common experimental configuration.

Setting	Value
Forecast horizons	6 (5, 10, 15, 20, 25, 30 minutes)
Split	Chronological 70/15/15
Transformer blocks	2
Token dimension	64
Attention heads	4
Feed-forward dimension	128
Dropout	0.1
Optimizer	Adam, learning rate 0.0003, epsilon 0.00001
Loss	Mean absolute error

Setting	Value
Epochs / batch size	150 / 256
Seeds	0, 1, 2, 3, 4
Activations	KANB17, KANB18, KANB21, ReLU, GELU, ParametricGELU

The frozen PEMS04 comparison used the same two-block feature-tokenized Transformer, with a separate training protocol for the denser road-traffic series. Adam used a learning rate of 3×10^{-4} and epsilon 10^{-5} with batch size 512. Training ran for at most 150 epochs, with validation-loss early stopping (patience 20, minimum change 10^{-4}) and restoration of the best validation weights. Six activations, five selected sensors and five seeds produced 150 runs. The sensor set and training settings were fixed before the test metrics were evaluated once. Table 3.3 records the full configuration.

Table 3.3. Frozen PEMS04 experimental configuration.

Setting	Value
Dataset	PEMS04 (PeMSD4), 1 January–28 February 2018
Raw tensor	16,992 time steps $\times$ 307 sensors $\times$ 3 measurements
Measurements / sampling	Total flow, average occupancy and average speed; 5-minute intervals
Selected sensors	2, 106, 202, 289 and 303
Forecast target	Traffic flow at 5, 10, 15, 20, 25 and 30 minutes
Inputs per sample	71 (36 target history, 24 neighbour flow, 6 seasonal lags, 5 calendar)
Split	Chronological 70/15/15
Samples per sensor	9,872 training; 2,543 validation; 2,543 test
Scaling	Inputs and per-horizon targets standardized from training data only
Transformer	Token dimension 64; 2 blocks; 4 heads; feed-forward dimension 128; dropout 0.1
Optimizer / loss	Adam; learning rate 3×10^{-4} ; epsilon 10^{-5} ; mean absolute error loss
Training control	Batch 512; maximum 150 epochs; early stopping patience 20; minimum change 10^{-4} ; restore best weights
Activations	KANB17, KANB18, KANB21, ReLU, GELU and ParametricGELU
Seeds / runs	5 seeds; 150 runs

4. Results and analysis

4.1 Main results on multi-horizon vessel-count forecasting

In the final results, we do not find one, singular, clear winner. The lowest mean validation relative MAE belongs to GELU with a score of 0.8037, followed by KANB21 (0.8045) and ParametricGELU (0.8050). On the testing part of the dataset, KANB17 has the lowest macro-average relative MAE (0.8498), but this is simply a descriptive result and not something that changes our selection decision. ReLU wins five of the ten series, having also the best average series rank (2.30). The rest of the five wins are shared between GELU, KANB17, KANB21 and ParametricGELU.

Performance per region differs much more than per activation. Across the four Aegean series, there is a roughly 37% improvement over persistence, having a mean test relative MAE of about 0.63 for every activation. In the six Brest series, activation means are closer to 1.00. It can be noticed that Brest 74768 and 75108 are persistence-dominated, while the rest of the series retain a positive skill.

A high-headroom case can be found in Aegean 19285, where KANB21 is the best activation with mean test MAE 1.2917 and a persistence of 2.2681, which results to a skill of 0.430. The horizon curves for all six activations remain very close to one another, indicating that most of the improvement can be attributed to the forecasting model itself instead of the choice of activation.

When the six forecast horizons are examined separately the KANB results become more specific. KANB21 comes on top among all six activations in 9 out of the 60 series-horizon cells, KANB17 in 7 and KANB18 in 6. Compared against the best conventional activations of each series and horizons, their corresponding counts are 15, 12 and 9. If we take the whole series into account, however, KANB21 wins Aegean 19285 and KANB17 wins Brest 74768, which still has a relative MAE of 1.5085. Even the best activation in this case, is worse than persistence.

Table 4.1. Activation ranks across the ten series (lower is better).

Activation	Average rank	Series wins
ReLU	2.30	5
GELU	3.40	2
ParametricGELU	3.60	1
KANB17	3.80	1
KANB21	3.90	1
KANB18	4.00	0

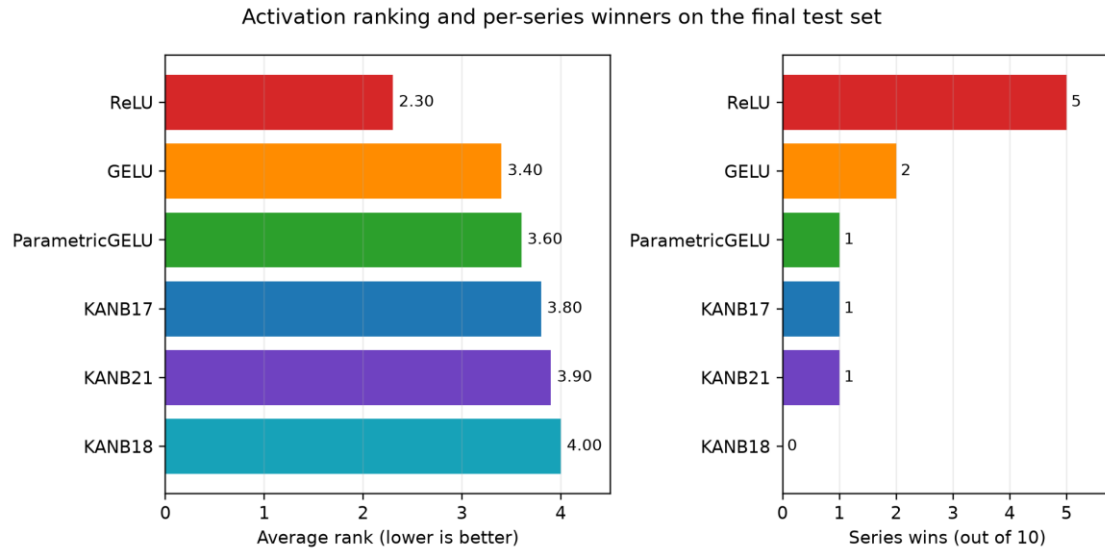


Figure 4.1. Average activation rank and number of series wins.

Table 4.2. Best neural activation and persistence-relative skill by series.

Series	Best activation	Best MAE	Persistence	Skill
Aegean 19285	KANB21	1.2917	2.2681	+0.430
Aegean 19653	ReLU	1.8892	2.8063	+0.327
Aegean 19837	ReLU	1.9098	3.3959	+0.438
Aegean 20017	GELU	1.7865	2.5079	+0.288
Brest 74429	GELU	0.1659	0.2032	+0.183
Brest 74767	ReLU	0.2206	0.2890	+0.237
Brest 74768	KANB17	0.6769	0.4487	-0.508
Brest 74769	ReLU	1.0978	1.3296	+0.174
Brest 75108	ParametricGELU	0.2454	0.2354	-0.043
Brest 75447	ReLU	0.1257	0.1455	+0.136

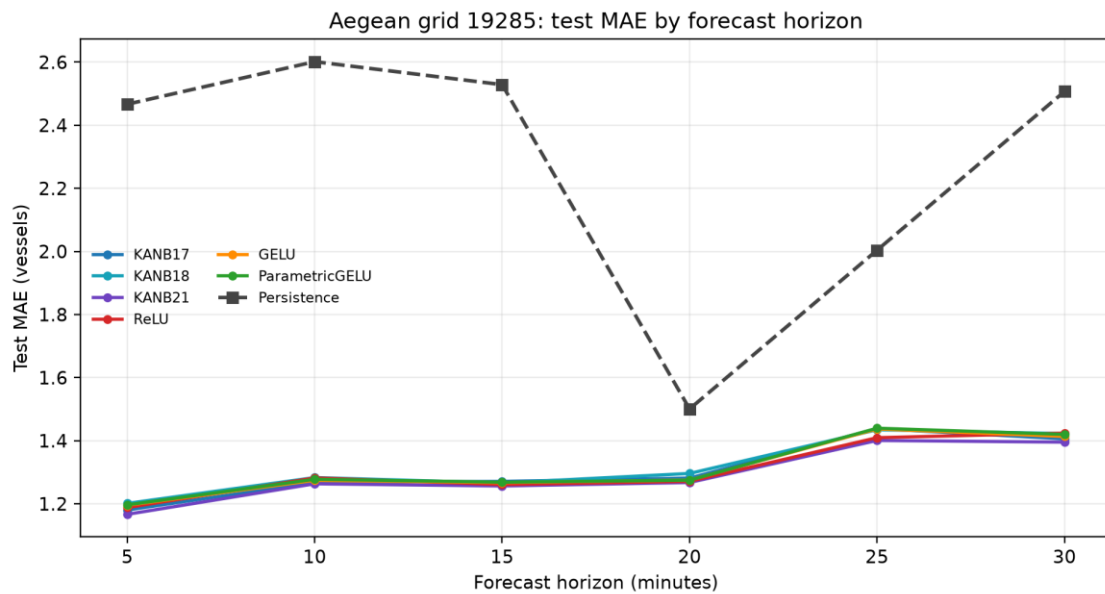


Figure 4.2. Test MAE by forecast horizon for Aegean grid 19285.



Figure 4.3. Skill relative to persistence for every activation and series.

4.2 Controlled activation comparisons and baseline checks

The paired analysis avoids giving more weight to cells with more vessels by using series-level differences in test relative MAE. The mean KANB21-GELU difference is -0.0012 with a 95% bootstrap interval around -0.0200 to 0.0137 and a sign-flip p-value of 0.912, meaning that KANB21 and GELU are practically tied. KANB21 is also lower in four out of the ten series and none of the nine KANB versus baseline confidence intervals excludes zero. KANB17 has the lowest macro-average test relative MAE. However, pairing it against ReLU does not show significant differences or losses, while also winning in two of the ten paired comparisons.

ParametricGELU stays close to GELU. Ridge regression or histogram gradient boosting results in the lowest MAE in five of the series, while the tuned Transformer, has the lowest MAE on the rest of the five series. The results are therefore mixed. In some cases, the more complex neural model is better, while simple, classical methods can perform better in others. This comparison does not aim to establish a new method for vessel-count forecasting, but to simply describe the activation behavior within the chosen Transformer, as well as serving as a sanity check to see whether or not the Transformer model adds to the predictive value.

Table 4.3. Series-level paired relative-MAE differences; negative values favour KANB.

KANB	Baseline	Mean relative difference	95% bootstrap CI	Exact p
KANB17	ReLU	-0.0089	[-0.0422, +0.0174]	0.609
KANB17	GELU	-0.0040	[-0.0290, +0.0139]	0.943
KANB17	ParametricGELU	-0.0014	[-0.0239, +0.0146]	0.957
KANB18	ReLU	+0.0058	[-0.0055, +0.0177]	0.422
KANB18	GELU	+0.0107	[-0.0100, +0.0360]	0.436
KANB18	ParametricGELU	+0.0132	[-0.0094, +0.0441]	0.518
KANB21	ReLU	-0.0061	[-0.0356, +0.0166]	0.750
KANB21	GELU	-0.0012	[-0.0200, +0.0137]	0.912
KANB21	ParametricGELU	+0.0013	[-0.0157, +0.0163]	0.889

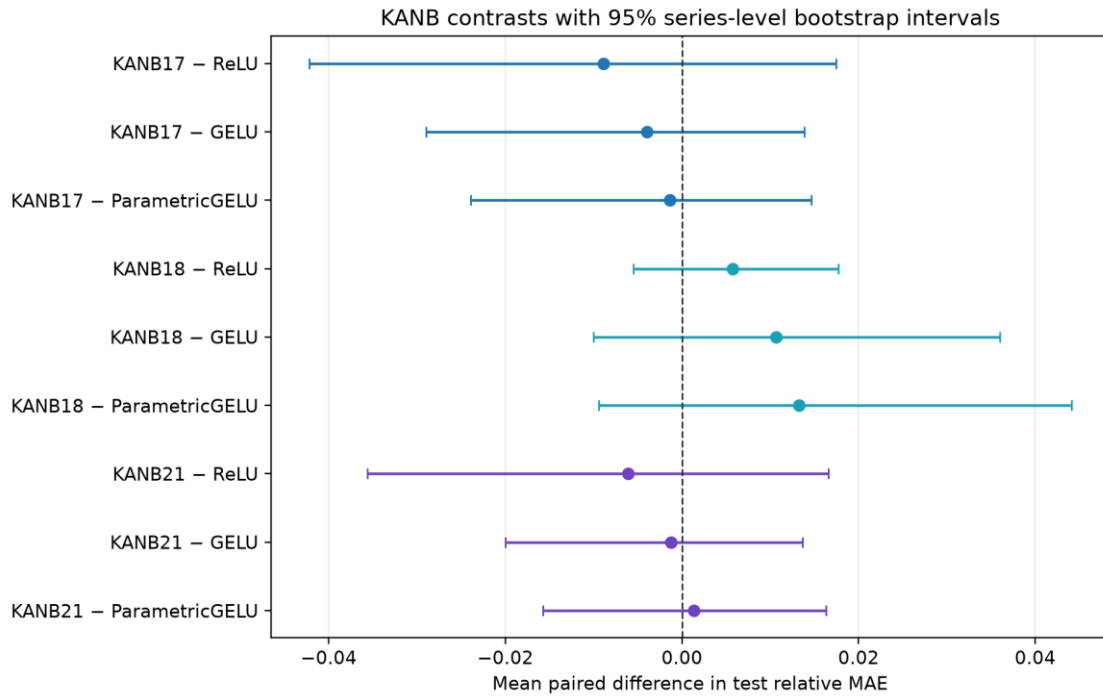


Figure 4.4. Paired KANB effects with series-level bootstrap confidence intervals.

Table 4.4. Best neural and classical sanity-check model by series.

Series	Best neural	MAE	Best classical	MAE
Aegean 19285	KANB21	1.2917	Hist Gradient Boosting	1.3057
Aegean 19653	ReLU	1.8892	Ridge	1.6698
Aegean 19837	ReLU	1.9098	Ridge	1.8561
Aegean 20017	GELU	1.7865	Ridge	1.7334
Brest 74429	GELU	0.1659	Ridge	0.2197
Brest 74767	ReLU	0.2206	Hist Gradient Boosting	0.3391
Brest 74768	KANB17	0.6769	Hist Gradient Boosting	0.6076
Brest 74769	ReLU	1.0978	Hist Gradient Boosting	1.0715
Brest 75108	ParametricGELU	0.2454	Ridge	0.3690
Brest 75447	ReLU	0.1257	Ridge	0.1768

4.3 Training dynamics and computational cost

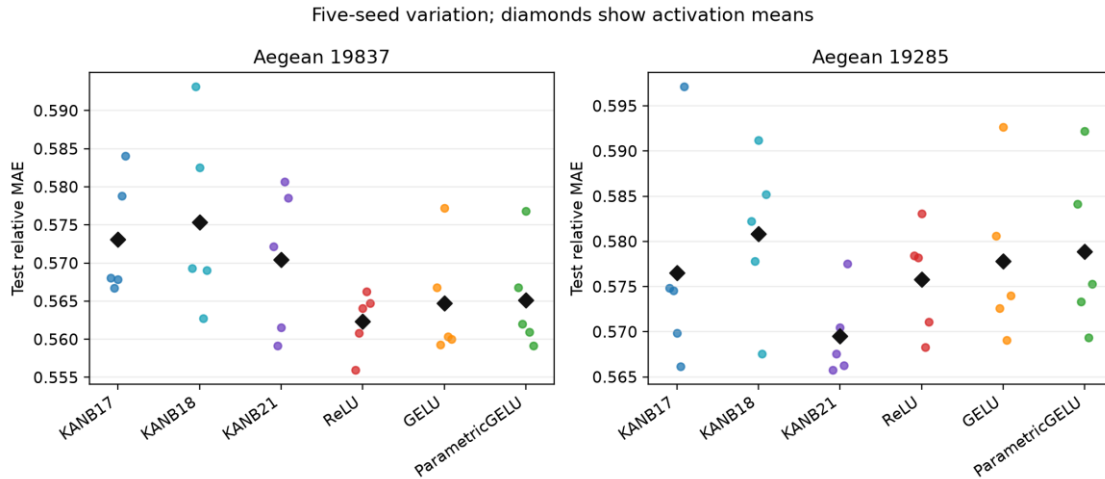
With all 300 runs completed successfully, the mean best validation epoch ranged from 106.2 to 116.0 for KANB18 and KANB17. Since early stopping was not used for the experiments, the protocol trains for 150 epochs and evaluates the last model state instead of restoring the best checkpoint. The mean final-minus-best validation-loss gap was similar across variations, ranging from 0.0195 to 0.0212.

Mean training time was relatively close for the 6 activations. It ranges from 162.1 seconds for ReLU, to 182.4 seconds for KANB18. GELU averages 165.2 seconds, ParametricGELU 172.2, KANB17 176.8 and KANB21 181.1. The trainable activations do add measurable runtime, even though their parameter count remained negligible relative to the Transformer.

Within each series, the five-seed results overlap. In some cases, the change between seeds is as substantial as the mean difference between activation families. Hence, the analysis retains all of the five seeds and reports paired averages instead of selecting a single run that was more favorable.

Table 4.5. Training-time and validation-loss diagnostics across 50 runs per activation.

Activation	Mean time (s)	Mean best epoch	Final-best val gap
KANB17	176.8	116.0	0.0202
KANB18	182.3	106.2	0.0212
KANB21	181.1	113.9	0.0195
ReLU	162.1	113.8	0.0198
GELU	165.2	110.8	0.0200
ParametricGELU	172.2	111.7	0.0199

**Figure 4.5. Seed variation and mean MAE on two high-headroom Aegean series.**

4.4 Learned activation curves

Before aggregation, the final KANB parameters were evaluated as functions. Figure 4.6 shows the mean curve of each activation based on the final 50 runs for that activation in each block. The band shows one standard deviation and faint lines are for individual fits. Since the relationship between parameters and functions is nonlinear, the curves were evaluated before averaging. KANB17 remains saturating while its a scale falls from 3.0 to 2.878 in block 1 and 2.807 in block 2. KANB18 keeps a linearly positive branch, while YP2 moves from -0.2 to -0.361 and -0.266 in the two blocks. KANB21 stays close to its initial form with the mean slope b being 0.950 in block 1 and 0.956 in block 2.

The fact that the parameters changed in both blocks, works as confirmation that the activation layers received gradients and were trained. The activations converge to clearly visible different shapes without producing big differences in relative MAE.

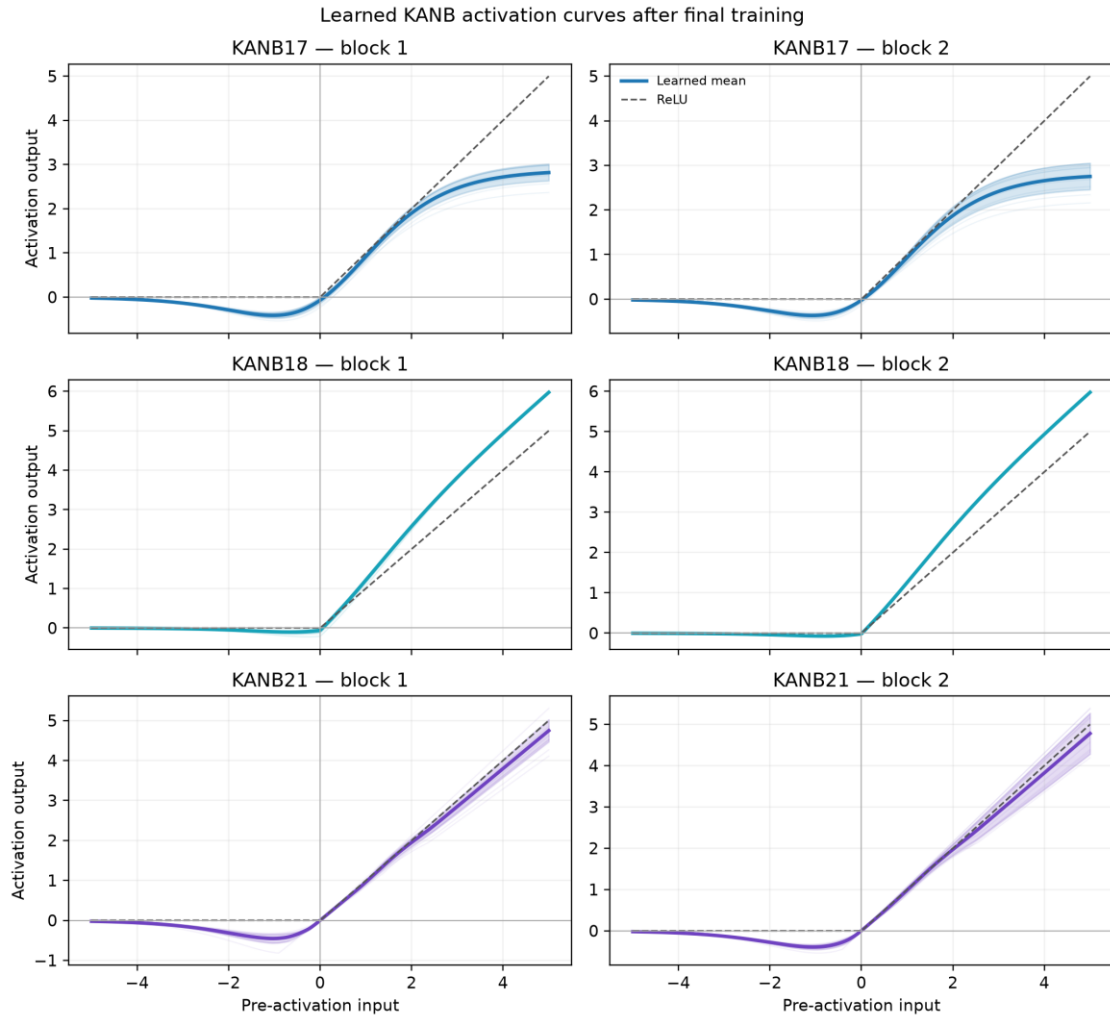


Figure 4.6. Learned KANB curves in Transformer blocks 1 and 2.

4.5 Follow-up training experiments

A few follow-up experiments were performed, to address two practical questions. Can a larger batch change the performance of KANB17? If the learning rate is moved away from a plateau change the learned activation parameters enough to improve validation error? The protocol contained 5 series with 5 seeds for 25 runs. All other settings were left the same as the initial experiment (150 epochs, depth 2, four heads, Adam epsilon 10^{-5} , initial learning rate 3×10^{-4}).

Table 4.6. Validation-only KANB17 training follow-up (lower is better).

Protocol	Runs	Mean validation relative MAE	Paired Δ
Batch 256, constant LR	25	0.6620	—
Batch 512, constant LR	25	0.6678	+0.0058
Batch 256, ReduceLROnPlateau	25	0.6891	+0.0270
Batch 256, plateau + warm restart	25	0.6757	+0.0137

The results showed that our original protocol was the best, with a mean validation relative MAE of 0.6620. Batch size 512 increases it to 0.6678, the warm-restart protocol to 0.6757, and ReduceLROnPlateau alone to 0.6891. The one-time restart was triggered in 22 of the 25 runs after the learning rate fell below 10^{-5} and reset it to 3×10^{-4} . Parameter changes immediately after a restart were 4.65 to 11.66 times larger than those following a learning-rate reduction. The intervention therefore moved the learned KANB17 curve away from the plateau, but that movement did not produce a lower validation error so the final configuration was left unchanged.

4.6 PEMS04 road-traffic experiment

4.6.1 Frozen protocol and aggregate results

As a final experiment, the PEMS04 dataset was chosen since, as a series, it is denser than the maritime cell counts, and the target sensors have specific graph neighbors. The question posed is whether the behavior of the activation functions changes on road traffic with a stronger and short-term structure.

Table 4.7 shows the exact performances of each activation function. The first comparison only used validation data while the training protocol and the sensor set were already frozen. As shown, KANB21 records the lowest mean relative MAE with a value of 0.743930. ReLU comes second with a MAE of 0.744528. Afterwards, the test partition was evaluated and KANB18 came first, showing an improvement over GELU of 0.357%.

Table 4.7. Aggregate PEMS04 relative MAE across the frozen validation comparison, frozen final test, and post-test optimized validation-only sensitivity analysis (25 sensor-seed runs per activation; lower is better).

Activation	Frozen validation	Frozen final test	Optimized validation only
KANB18	0.745400	0.747688	0.736586
ReLU	0.744528	0.749232	0.737207
KANB21	0.743930	0.749639	0.738261
ParametricGELU	0.745595	0.749852	0.736688
GELU	0.747169	0.750365	0.736629
KANB17	0.745240	0.751259	0.737249

PEMS04 activation rankings across the three reported comparisons

Mean relative MAE; lower is better

Frozen validation



Frozen final test



Post-test optimized validation only



Separate zoomed scale in each panel

Figure 4.7. Aggregate PEMS04 relative MAE across the frozen validation comparison, the one-time frozen final test, and the post-test optimized validation-only sensitivity analysis. Lower values are better; each panel uses a separate zoomed axis.

4.6.2 Sensor- and horizon-level results

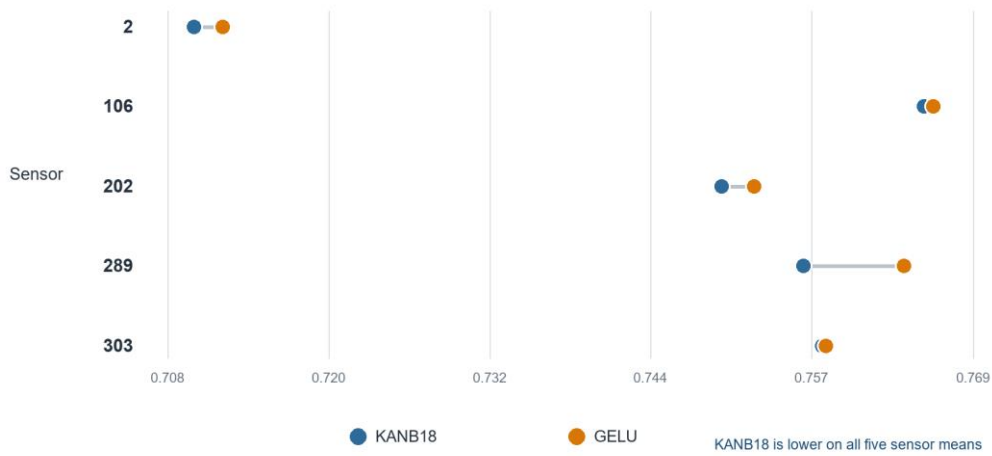
Table 4.8 and Figure 4.8 show the sensor-level comparison. KANB18 had a lower error rate on all five sensor means with Sensor 289 showing the clearest difference of 1.007%. For the paired sensor-seed comparison, KANB18 performed better in 16 out of 25 runs. With only five independent sensor means, an exact two-sided sign test gives $p = 0.0625$. The results seem to point to a certain direction, but the sample is not big enough for a strong claim.

Table 4.8. Sensor-level PEMS04 final-test relative MAE for KANB18 and GELU (five-seed means; negative delta favours KANB18).

Sensor	KANB18	GELU	Delta	Improvement (%)
2	0.709658	0.711867	-0.002208	0.310%
106	0.765275	0.766029	-0.000754	0.098%
202	0.749883	0.752332	-0.002449	0.326%
289	0.756095	0.763786	-0.007691	1.007%
303	0.757527	0.757810	-0.000283	0.037%

KANB18 versus GELU on each selected PEMS04 sensor

Mean final-test relative MAE; lower is better

**Figure 4.8. Paired sensor-level mean final-test relative MAE for KANB18 and GELU. KANB18 is lower on all five selected sensors, although the absolute differences remain small.**

When the horizons are broken down, something similar happens. Even though GELU performs slightly better at 5 minutes, KANB18 surpasses it in the timeframe of 10 to 30 minutes. The gap is still modest, with the highest value at 25 minutes (0.005161 relative MAE).

Table 4.9. PEMS04 final-test relative MAE by forecast horizon for KANB18 and GELU (25-run means; negative delta favours KANB18).

Horizon (min)	KANB18	GELU	Delta	Lower activation
5	0.781126	0.779672	+0.001454	GELU
10	0.772984	0.773877	-0.000893	KANB18
15	0.749928	0.753242	-0.003314	KANB18
20	0.742534	0.745718	-0.003184	KANB18
25	0.726790	0.731951	-0.005161	KANB18
30	0.721542	0.725724	-0.004182	KANB18

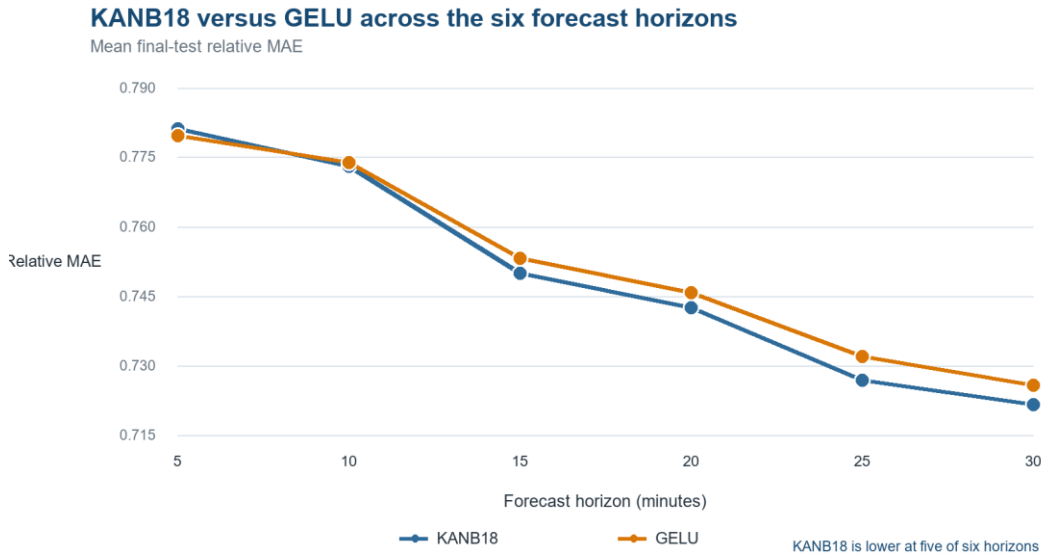


Figure 4.9. Mean final-test relative MAE by forecast horizon for KANB18 and GELU. KANB18 is lower at five of the six horizons, while GELU is lower at the 5-minute horizon.

4.6.3 Post-test validation sensitivity

After the frozen test had been evaluated, a validation-only sweep compared five training protocols over three seeds. The shared protocol selected across KANB17, KANB18 and KANB21 used batch size 256, an initial learning rate of 10^{-3} and ReduceLROnPlateau. A full six-activation validation comparison with five seeds then placed KANB18 first at 0.736586 and GELU second at 0.736629, a difference of only 0.000043. The ranking therefore remained favorable to KANB18, but the near-tie also shows how little the training change separated the activations. No test metrics were used in this sensitivity analysis; its values appear in the last column of Table 4.7 and the final panel of Figure 4.7.

Conclusions

Summary of findings

This thesis tested KANB17, KANB18 and KANB21 in a feature-tokenized Transformer for multi-horizon mobility forecasting. For the main maritime experiment, the hyperparameters were selected after experimentation and based on validation data. The six activations were then evaluated on ten Aegean and Brest series using five seeds on each run. The final result contains 300 successful runs while keeping the test split separate from tuning. A separate PEMS04 experiment added five road traffic sensors and 150 runs, again, using hyperparameter tuning based on training and validation data only.

On the maritime data, GELU gives the strongest validation score, KANB17 the lowest macro-average test relative MAE, and ReLU the best average series rank. These different summaries do not point to a single dominant activation. In the most relevant tuned comparison, KANB21 and GELU are essentially tied on the test set, and none of the paired KANB contrasts is statistically supported. Under this protocol, KANB is competitive but not consistently superior. At a finer level, all three KANB variants win some series-horizon cells, but only KANB17 and KANB21 win a complete series. The KANB17 training follow-up also leaves the original batch-256 constant-learning-rate protocol as the strongest validation setting. On the PEMS04 data, KANB18 had a slightly more favorable pattern, achieving the lowest mean relative MAE and improving on GELU by 0.357%.

In conclusion, the experiments cannot establish the KANB as a superior activation function in mobility forecasting. They did show, however, that KANB managed to remain competitive in two different mobility forecasting settings, while even having a slight edge for the PEMS04 dataset. This shows that when the data are more dense and have a stronger short-term seasonal structure, the trainable activations with the flexibility to adapt, will have room for improvement over the classical ones. Experiments that use more sensors, nonlinear datasets or shared regions, could assist in determining whether these small improvements detected here, can become larger when the variety and the amount of data increases.

Limitations

The results are based on ten maritime grid-cell series and five of the PEMS04 sensors, all of which were assessed using the same feature-tokenized Transformer architecture. Since attention mechanisms act across the input variables rather than across timestamps, the findings cannot be directly extended to temporal-token Transformers, recurrent models or graph neural networks.

In the case of the PEMS04 experiment, the flow data from neighboring sensors were included as standard input features; the model did not learn a graph representation. The individual grid cells or traffic sensors were trained separately, which meant that the models were unable to learn any common regional patterns. Moreover, the PEMS04 subset was rather small. The five sensors were chosen from the 307-sensor dataset by means of screening that involved only the training and validation data. As a result, the test data was excluded from the selection process, but it is possible that the performance on these particular sensors does not reflect the entire road network. Since the PEMS04 observations only include the months of January and February 2018, seasonal generalization is also further limited.

The training protocol was not the same in the two data settings. In the case of the maritime comparison a fixed schedule of 150 epochs was used without early stopping, which meant that the final weights did not have to be those from the epoch that had the lowest validation loss. For PEMS04, early stopping based on validation loss was employed and the best weights were restored. This difference makes it more difficult to attribute the change in activation rankings solely to the datasets.

The maritime scheduler and the warm-restart checks, as well as the later batch size and learning rate sweep in PEMS04, all made use of the same development data and must therefore be regarded as sensitivity checks not as independent confirmations.

While using five seeds reduces reliance on a single initialization, the number of independent series is still restricted. Specifically, the PEMS04 comparison includes only five sensor means. Even though KANB18 was lower than GELU for all five of them, the exact two-sided sign test yielded a p-value of 0.0625 and the sensor-level confidence interval did include zero. The observed improvement must therefore be regarded as suggestive not conclusive.

Since a frozen-KANB control was not included, the study is also unable to distinguish between the effect of trainability and the effect of the variants' initial functional forms. In the end, Brest 74768 and Brest 75108 were both dominated by persistence regardless of the activation functions. Changes in distribution were also observed between the chronologically ordered splits. Their failure cannot be blamed on the activation alone since a model trained on one vessel-count scenario might have difficulty when the level and frequency of activity alter during the testing period.

Directions for future work

The most useful next comparison would be a recurrent or temporal-token model, such as an LSTM, trained on the same splits and evaluated with the same persistence-relative metrics. That would test whether explicit sequential structure matters more than the activation choice. Checkpoint restoration, regional models with shared parameters, graph connections between neighboring cells and count-aware probabilistic losses are also reasonable extensions. Within the completed experiment, however, the defensible conclusion remains narrow: KANB is a plausible low-parameter replacement for fixed activations, with benefits that depend on the individual series.

Glossary of Terms

Foreign term	Greek term
Activation function	Συνάρτηση ενεργοποίησης
Adaptive / trainable activation	Προσαρμοστική / εκπαιδεύσιμη ενεργοποίηση
Automatic Identification System	Αυτόματο Σύστημα Αναγνώρισης
Attention	Μηχανισμός προσοχής
Feature token	Token γνωρίσματος
Forecast horizon	Χρονικός ορίζοντας πρόβλεψης
Grid cell	Γεωγραφικό κελί
Mean Absolute Error	Μέσο απόλυτο σφάλμα
Mobility forecasting	Πρόβλεψη κινητικότητας
Persistence forecast	Πρόβλεψη επιμονής
Random seed	Τυχαίος σπόρος
Self-attention	Αυτοπροσοχή
Time lag	Χρονική υστέρηση
Vessel-count forecasting	Πρόβλεψη αριθμού πλοίων

List of Abbreviations and Acronyms

AIS — Automatic Identification System

FFN — Feed-Forward Network

GELU — Gaussian Error Linear Unit

GNN — Graph Neural Network

KAN — Kolmogorov–Arnold Network

KANB — Kolmogorov–Arnold Network Bézier activation family

LSTM — Long Short-Term Memory

MAE — Mean Absolute Error

MLP — Multilayer Perceptron

ReLU — Rectified Linear Unit

RNN — Recurrent Neural Network

SD — Standard Deviation

Bibliography

- [1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- [2] Cheng, J., Li, Q., Lin, T., & Shen, Z. (2025). Interpolation, approximation, and controllability of deep neural networks. *SIAM Journal on Control and Optimization*, 63(1), 625–649.
- [3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778).
- [4] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179–211.
- [5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- [6] Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207, Art. no. 117921.
- [7] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)* (pp. 1263–1272).
- [8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.
- [9] Lee, J. K. (2025). Transformer-Based Vehicle-Trajectory Prediction at Urban Low-Speed T-intersection. *Sensors*, 25(14), 4256. <https://doi.org/10.3390/s25144256>.
- [10] Emanuel, R. H. K., Docherty, P. D., Lunt, H., & Möller, K. (2024). The effect of activation functions on accuracy, convergence speed, and misclassification confidence in CNN text classification: A comprehensive exploration. *Journal of Supercomputing*, 80(1), 292–312. <https://doi.org/10.1007/s11227-023-05441-7>.
- [11] Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, 138, 14–32. <https://doi.org/10.1016/j.neunet.2021.01.026>.
- [12] Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503, 92–108. <https://doi.org/10.1016/j.neucom.2022.06.111>.
- [13] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
- [14] Joudaki, A., Daneshmand, H., & Bach, F. (2023). On the impact of activation and normalization in obtaining isometric embeddings at initialization. *Advances in Neural Information Processing Systems*, 36, 39855–39875.
- [15] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 2672–2680).
- [16] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)* (pp. 6840–6851).
- [17] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., & Tegmark, M. (2024). KAN: Kolmogorov-Arnold networks. *arXiv preprint arXiv:2404.19756*.
- [18] Kanakoudis, G., Mandalis, P., & Pelekis, N. (2025). KANB: A Family of Parametric Activation Functions With Trainable Parameters. *IEEE Access*, 13, 146081–146087. <https://doi.org/10.1109/ACCESS.2025.3600484>.
- [19] Evmidis, N., Michaelides, M. P., & Herodotou, H. (2025). Vessel Trajectory Prediction with Deep Learning: Temporal Modeling and Operational Implications. *Journal of Marine Science and Engineering*, 13(8), 1439. <https://doi.org/10.3390/jmse13081439>.

- [20] Liu, R. W., Zheng, W., & Liang, M. (2024). Spatio-temporal multi-graph transformer network for joint prediction of multiple vessel trajectories. *Engineering Applications of Artificial Intelligence*, 129, 107625. <https://doi.org/10.1016/j.engappai.2023.107625>.
- [21] Mandalis, P., Chondrodima, E., Kontoulis, Y., Pelekis, N., & Theodoridis, Y. (2025). A transformer-based method for vessel traffic flow forecasting. *Geoinformatica*, 29(1), 149–173. <https://doi.org/10.1007/s10707-024-00521-z>.
- [22] Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., & Theodoridis, Y. (2017). Online event recognition from moving vessel trajectories. *Geoinformatica*, 21(2), 389–427. <https://doi.org/10.1007/s10707-016-0266-x>.
- [23] Bao, K., & Wang, P. (2024). Artificial Intelligence in Ship Trajectory Prediction. *Journal of Marine Science and Engineering*, 12(5), 769. <https://doi.org/10.3390/jmse12050769>.
- [24] Vouros, G. A., Andrienko, G., Doukeridis, C., Pelekis, N., Artikis, A., Joussemme, A.-L., Ray, C., Cordero, J. M., & Scarlatti, D. (Eds.). (2020). *Big Data Analytics for Time-Critical Mobility Forecasting: From Raw Data to Trajectory-Oriented Mobility Analytics in the Aviation and Maritime Domains*. Springer. ISBN 978-3-030-45163-9.
- [25] Xu, K., Chen, L., & Wang, S. (2024). Kolmogorov-Arnold Networks for Time Series: Bridging Predictive Power and Interpretability. *arXiv preprint arXiv:2406.02496*.
- [26] Chen, C., Petty, K., Skabardonis, A., Varaiya, P., & Jia, Z. (2001). Freeway Performance Measurement System: Mining Loop Detector Data. *Transportation Research Record*, 1748(1), 96–102. <https://doi.org/10.3141/1748-12>.
- [27] Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019). Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), 922–929. <https://doi.org/10.1609/aaai.v33i01.3301922>.

Appendices

Appendix A — KANB implementation excerpt

The listing below reproduces the activation equations used in the experiment. Parameters are scalar TensorFlow weights with the constraint $[-20, 20]$. Each Transformer block receives a separate activation-layer instance.

```
def _kanb_cubic(t, yp1, yp2):
    return 3*(1-t)**2*t*yp1 + 3*(1-t)*t**2*yp2 + t**3

class KANB17Activation(tf.keras.layers.Layer):
    def call(self, x):
        t = tf.math.sigmoid(x)
        quad = 2*t*(1-t)*self.yp1 + t**2
        return self.a * quad

class KANB18Activation(tf.keras.layers.Layer):
    def call(self, x):
        t = tf.math.sigmoid(x)
        return _kanb_cubic(t, self.yp1, self.yp2) + tf.nn.relu(x)

class KANB21Activation(tf.keras.layers.Layer):
    def call(self, x):
        t = tf.math.sigmoid(x)
        quad = 2*t*(1-t)*self.yp1 + t**2
        return tf.maximum(self.a * quad, self.b * x)
```

Appendix B — Detailed maritime activation-comparison results

Table B.1 contains the mean test MAE over the six horizons and five seeds for the ten Aegean and Brest series.

Table B.1. Mean six-horizon test MAE by activation.

Series	KANB17	KANB18	KANB21	ReLU	GELU	ParamGELU
Aegean 19285	1.3076	1.3173	1.2917	1.3060	1.3105	1.3129
Aegean 19653	1.9264	1.8918	1.8924	1.8892	1.9238	1.9191
Aegean 19837	1.9462	1.9538	1.9371	1.9098	1.9177	1.9191
Aegean 20017	1.7940	1.7870	1.7924	1.7898	1.7865	1.7882
Brest 74429	0.1745	0.1768	0.1737	0.1672	0.1659	0.1683
Brest 74767	0.2212	0.2213	0.2219	0.2206	0.2230	0.2216
Brest 74768	0.6769	0.7032	0.6904	0.7119	0.7233	0.7175
Brest 74769	1.1468	1.1223	1.1494	1.0978	1.1320	1.1531
Brest 75108	0.2528	0.2768	0.2551	0.2821	0.2533	0.2454
Brest 75447	0.1276	0.1277	0.1289	0.1257	0.1264	0.1262

Appendix C — Hyperparameter grids

In the maritime experiments, the hyperparameters were tuned one by one using KANB21 on five development series and with the seeds 0, 1 and 2. Throughout the stages, only one factor would change each time, and would then be carried onto the next stage. Table C.1 shows the values considered and the configuration.

Table C.1. Sequential tuning grid and selected final configuration.

Factor	Evaluated and selected value(s)
Tuning activation	KANB21
Development series	Aegean 19285, 19653, 19837, 20017; Brest 74769
Tuning seeds	0; 1; 2
Learning rate	0.0001; 0.0003; 0.001; 0.003 → selected 0.0003
Adam epsilon	10^{-8} ; 10^{-7} ; 10^{-6} ; 10^{-5} ; 10^{-4} → selected 10^{-5}
Batch size	128; 256; 512; 1024 → selected 256
Epochs	75; 100; 150; 200; 300 → selected 150
Early stopping	disabled; patience 10; 20; 30 → selected disabled
Transformer depth	1; 2; 3; 4 → selected 2
Attention heads	2; 4; 8 → selected 4
Final seeds	0; 1; 2; 3; 4
Final activations	KANB17; KANB18; KANB21; ReLU; GELU; ParametricGELU
Trainable activation-parameter constraint	[-20, 20]
KANB17 initialization	YP1 = -0.5; a = 3.0 (unchanged)
KANB18 initialization	YP1 = -0.1; YP2 = -0.2 (unchanged)
KANB21 initialization	YP1 = -0.5; a = 3.0; b = 1.0 (unchanged)
ParametricGELU initialization	a = 1.0; b = 1.0

PEMS04 used a separate frozen training protocol. Table C.2 distinguishes the configuration evaluated once on the test set from the later sensitivity sweep, which remained validation-only. The sensor set was fixed at 2, 106, 202, 289 and 303.

Table C.2. PEMS04 frozen protocol and post-test validation-only training sweep.

Protocol	Training settings	Evaluation scope
Frozen validation/final-test	Batch 512; initial LR 3×10^{-4} ; constant LR; maximum 150 epochs; early-stopping patience 20.	Five sensors × six activations × five seeds. The test set was evaluated once in the final 150-run experiment.
control_b512_lr3e4	Batch 512; initial LR 3×10^{-4} ; constant LR.	Three KANBs × five sensors × three seeds (45 runs); validation only.

Protocol	Training settings	Evaluation scope
b256_lr3e4_plateau	Batch 256; initial LR 3×10^{-4} ; ReduceLROnPlateau.	Three KANBs $\times$ five sensors $\times$ three seeds (45 runs); validation only.
b256_lr6e4_plateau	Batch 256; initial LR 6×10^{-4} ; ReduceLROnPlateau.	Three KANBs $\times$ five sensors $\times$ three seeds (45 runs); validation only.
b256_lr1e3_plateau (selected shared)	Batch 256; initial LR 1×10^{-3} ; ReduceLROnPlateau.	Three KANBs $\times$ five sensors $\times$ three seeds (45 runs); validation only.
b128_lr3e4_plateau	Batch 128; initial LR 3×10^{-4} ; ReduceLROnPlateau.	Three KANBs $\times$ five sensors $\times$ three seeds (45 runs); validation only.

For PEMS04, the selected protocols used Adam with epsilon 10^{-5} and restored the best validation weights. There was a maximum of 180 epochs along with validation-loss early stopping with patience 30 and minimum change 10^{-4} . The plateau schedules used a reduction factor of 0.5, patience 8 and a minimum learning rate of 10^{-5} . The 225-run sweep was conducted after the frozen test and did not evaluate the test set.