National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science

Analysis of Deep Federated Learning on Early Prediction of ICU Mortality Risk

Diploma Thesis

of

ATHANASIOS GEORGOUTSOS

Supervisor: Vassiliki Kantere
Assistant Professor, NTUA

Athens, July 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science

Analysis of Deep Federated Learning on Early Prediction of ICU Mortality Risk

Diploma Thesis

of

ATHANASIOS GEORGOUTSOS

Supervisor: Vassiliki Kantere
Assistant Professor, NTUA

Approved by the examination committee on 4th July 2023.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Vassiliki Kantere	Dimitrios Tsoumakos	Symeon Papavassiliou
Assistant Professor, NTUA	Associate Professor, NTUA	Professor, NTUA

Athens, July 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science

(Signature)

.....
Athanasios Georgoutsos

Electrical & Computer Engineer Graduate, NTUA

Περίληψη

Τα τελευταία χρόνια, τεράστιες ποσότητες δεδομένων παράγονται με ραγδαίο ρυθμό, περιέχοντας πληροφορίες που θα μπορούσαν να βελτιώσουν σημαντικά διάφορες πτυχές της καθημερινής ζωής, εάν αξιοποιηθούν αποτελεσματικά με τεχνικές μηχανικής μάθησης. Ωστόσο, η κεντρική μηχανική μάθηση συχνά υπόκειται σε προβλήματα ιδιωτικότητας και ασφάλειας, που εμποδίζουν την ανάπτυξη ισχυρών μοντέλων. Η ομοσπονδιακή μάθηση είναι μια νέα προσέγγιση στο πεδίο της μηχανικής μάθησης, η οποία αντιμετωπίζει αυτά τα ζητήματα επιτρέποντας σε διάφορους φορείς να εκπαιδεύσουν συνεργατικά έναν αλγόριθμο χωρίς την ανταλλαγή των τοπικών τους δεδομένων. Μεταξύ άλλων, ο τομέας της υγείας θα μπορούσε να επωφεληθεί σημαντικά από την ομοσπονδιακή μάθηση, καθώς τα δεδομένα από διάφορα κέντρα υγείας δεν μπορούν να συγκεντρωθούν λόγω του απορρήτου των ασθενών.

Στην παρούσα διπλωματική εργασία, μελετάμε τη χρήση προηγμένων αλγορίθμων ομοσπονδιακής μάθησης για την έγκαιρη πρόβλεψη του κινδύνου θνησιμότητας στη ΜΕΘ. Συγκεκριμένα, επικεντρωνόμαστε στη χρήση των μετρήσεων ζωτικών ενδείξεων και των εργαστηριακών εξετάσεων, σε μορφή πολυμεταβλητών χρονοσειρών, από τις πρώτες 24 ώρες παραμονής στη ΜΕΘ για να εκτιμήσουμε το ρίσκο θνησιμότητας κατά τις επόμενες 48 ώρες. Τα δεδομένα μας προέρχονται από νοσοκομεία των ΗΠΑ μεταξύ 2014 και 2015, επιτρέποντάς μας να κατασκευάσουμε ένα ρεαλιστικό ομοσπονδιακό περιβάλλον. Για να αποκτήσουμε μια καλύτερη κατανόηση της ομοσπονδιακής μάθησης, σε σχέση με αυτό το πρόβλημα, σχεδιάζουμε μια σειρά πειραματικών σεναρίων. Στόχος μας είναι να εξερευνήσουμε τη γενικευσιμότητα και την ενσωμάτωση διαφορετικών μοντέλων αναδρομικών νευρωνικών δικτύων βαθιάς μάθησης στο πλαίσιο της ομοσπονδιακής μάθησης, την ευαισθησία διαφορετικών αλγορίθμων σε ανομοιόμορφες κατανομές των τοπικών δεδομένων και τη συνεισφορά συγκεκριμένων νοσοκομείων στην ανάπτυξη ενός μοντέλου ομοσπονδιακής μάθησης.

Τα αποτελέσματά μας υποδεικνύουν ότι τα μοντέλα ομοσπονδιακής μάθησης παρουσιάζουν καλύτερες επιδόσεις από τα τοπικά μοντέλα και ελαφρώς χειρότερες επιδόσεις από τα μοντέλα κεντρικής μηχανικής μάθησης. Στη συνέχεια, ανάλογα με το μέγεθος και την κατανομή των δεδομένων των συνεργαζόμενων φορέων, ορισμένοι αλγόριθμοι ομοσπονδιακής μάθησης είναι πιο αποτελεσματικοί σε συγκεκριμένα σενάρια. Επιπλέον, νοσοκομεία διαφορετικού μεγέθους μπορούν να συμβάλλουν στη βελτίωση της επίδοση των προγνωστικών μοντέλων πάνω στα τοπικά δεδομένα τους, συμμετέχοντας στην εκπαίδευσή τους.

Λέξεις Κλειδιά

Ομοσπονδιακή Μάθηση, Βαθιά Νευρωνικά Δίκτυα, Αναδρομικά Νευρωνικά Δίκτυα, Μηχανική Μάθηση, Ιδιωτικότητα, Πρόβλεψη Θνησιμότητας, Εκτίμηση Κινδύνου, Χρονοσειρές

Abstract

In the technological age, vast amounts of data are being generated continuously, holding information that could significantly improve various aspects of everyday life, if effectively utilized with machine learning techniques. However, traditional centralized machine learning approaches are subject to privacy and security concerns, which impede the development of robust, generalizable models. Federated learning is a novel approach in the field of machine learning, which addresses the aforementioned issues by allowing multiple parties to collaboratively train a machine learning algorithm without exchanging their local data. One domain that could be significantly benefited from the advancement of federated learning is healthcare, where data from multiple medical institutions cannot be centralized due to patient privacy.

In this thesis, we investigate the application of state-of-the-art federated learning (FL) algorithms on the early prediction of ICU mortality risk. More specifically, we focus on utilizing the rich temporal dynamics of vital signs and laboratory results, in the form of multivariate time series, from the first 24 hours of an ICU stay to estimate the mortality risk in the following 48 hours. Our data originates from multiple hospitals across the US between 2014 and 2015, therefore allowing us to recreate a realistic, multi-center federated environment. In order to gain an insight into the FL framework, with regard to this task, we design a series of experimental scenarios. Our aim is to explore the generalizability and integration of different deep recurrent neural network (RNN) models with FL, the sensitivity of different FL algorithms in the presence of heterogeneous local data distributions and the effect of individual hospitals on the development of a FL model.

We compare our FL models with reference to the centralized machine learning (CML) and local machine learning (LML) approaches. Our results indicate that, in settings with non-IID datasets, the FL models are superior to the privacy-preserving LML models, in terms of AUROC, AUPRC and F1-Score, while they perform slightly worse, in general, than the CML models. Then, depending on the characteristics of the FL participants, in relation to data size and class representation, certain FL algorithms exhibit better suitability for specific scenarios. Moreover, hospitals with both smaller and larger datasets may improve the models' performance on their local data, by participating in FL model training.

Keywords

Federated Learning, Deep Neural Networks, Recurrent Neural Networks, Machine Learning, Privacy, Mortality Prediction, Risk Prediction, Time Series

Ευχαριστίες

Πρώτα, θα ήθελα να ευχαριστήσω θερμά την επιβλέπουσα καθηγήτριά μου, κα. Βασιλική Καντερέ, για την άψογη συνεργασία μας κατά τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας. Η διαρκής καθοδήγησή της και οι συμβουλές της ήταν καθοριστικές, τόσο για την πορεία της διπλωματικής μου όσο και για την ανάπτυξη του επιστημονικού τρόπου σκέψης μου και των γνώσεών μου. Οφείλω επίσης να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Παρασκευά Κερασιώτη για την πολύτιμη βοήθειά του και τις προτάσεις του σχετικά με τις ερευνητικές κατευθύνσεις που θα μπορούσα να ακολουθήσω σε αυτήν την εργασία.

Έπειτα, ευχαριστώ τους φίλους μου, οι οποίοι ήταν πάντα δίπλα μου κατά τη διάρκεια των αυτών των φοιτητικών χρόνων. Οι στιγμές που ζήσαμε μαζί μου έδωσαν την απαραίτητη ώθηση να συνεχίσω και με βοήθησαν να ξεπεράσω όποιες δυσκολίες αντιμετώπισα. Θα ήθελα να ευχαριστήσω ιδιαίτερα τον Σπύρο, με τον οποίο ξεκινήσαμε παρέα αυτό το ακαδημαϊκό ταξίδι, καθώς δεν θα μπορούσα να ανταποκριθώ στις απαιτήσεις των σπουδών με την ίδια επιτυχία δίχως τη συνεργασία μαζί του.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, Δημήτρη και Ελένη, και την αδερφή μου, Ιωάννα-Σοφία, και να εκφράσω τη βαθύτατη ευγνωμοσύνη μου για όλα όσα μου έχουν προσφέρει καθ' όλη τη διάρκεια της ζωής μου. Η αμείωτη στήριξή τους και η φροντίδα που μου παρείχαν συνέβαλε ουσιαστικά στην επιτυχημένη ολοκλήρωση αυτού του κύκλου των σπουδών μου, καθώς και στη διαμόρφωση του χαρακτήρα μου και στη συνεχή βελτίωσή μου ως άνθρωπος.

Αθήνα, 4η Ιουλίου 2023

Αθανάσιος Γεωργούτσος

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Εκτεταμένη Περίληψη στα Ελληνικά

Το κεφάλαιο αυτό αποτελεί μια εκτεταμένη περίληψη της παρούσας διπλωματικής εργασίας στα ελληνικά, όπου διατυπώνονται οι κεντρικές ιδέες από κάθε ενότητα της εργασίας με συνοπτικό τρόπο.

0.1 Εισαγωγή

Στη σημερινή ψηφιακή εποχή, η εκρηκτική αύξηση της παραγωγής δεδομένων από πληθώρα ηλεκτρονικών συσκευών έχει δημιουργήσει την ανάγκη για καλύτερη διαχείριση και αξιοποίησή τους σε διάφορους τομείς. Ένας από αυτούς είναι ο τομέας της υγείας, όπου η συνεχής καταγραφή ιατρικών πληροφοριών και μετρήσεων, με τη μορφή χρονοσειρών, σε ηλεκτρονική μορφή, προσφέρει τη δυνατότητα για δραματική βελτίωση της φροντίδας των ασθενών, της λήψης ιατρικών αποφάσεων και της έρευνας στο κομμάτι της υγείας. Ωστόσο, οι συνηθισμένες προσεγγίσεις αξιοποίησης δεδομένων από διαφορετικές πηγές προϋποθέτουν την συγκέντρωση των δεδομένων σε ένα κεντρικό μηχάνημα. Αυτό συνοδεύεται από διάφορα ζητήματα, όπως αυξημένες υπολογιστικές απαιτήσεις για το κεντρικό μηχάνημα, καθυστερήσεις και διακοπές λόγω δικτυακών προβλημάτων κατά τη μεταφορά των δεδομένων και θέματα ασφάλειας και ιδιωτικότητας ευαίσθητων δεδομένων. Το τελευταίο ζήτημα είναι μείζονος σημασίας για τον χώρο της υγείας, καθώς η επεξεργασία προσωπικών δεδομένων υπόκειται σε νομικές και ηθικές υποχρεώσεις, όπως εκείνες του Γενικού Κανονισμού για την Προστασία των Δεδομένων (General Data Protection Regulation - GDPR) της Ευρωπαϊκής Ένωσης [16], οι οποίες αποτρέπουν τη συγκέντρωση δεδομένων από διάφορα ιατρικά κέντρα για την εκπαίδευση ισχυρών προγνωστικών μοντέλων μηχανικής μάθησης. Η ομοσπονδιακή μάθηση (Federated Learning), μια νέα τεχνική μηχανικής μάθησης που επιτρέπει τη συνεργατική εκπαίδευση ενός μοντέλου από πολλαπλούς συνεργαζόμενους φορείς χωρίς τη συγκέντρωση των δεδομένων εκπαίδευσης, αποτελεί μια καινοτόμο, υποσχόμενη λύση για το παραπάνω πρόβλημα.

Στη παρούσα εργασία, εξετάζουμε το πρόβλημα της έγκαιρης πρόβλεψης του κινδύνου θνησιμότητας στις μονάδες εντατικής θεραπείας (ΜΕΘ) στο πλαίσιο της ομοσπονδιακής μάθησης. Η έγκαιρη πρόβλεψη του κινδύνου θνησιμότητας είναι μία από τα σημαντικότερες προκλήσεις στον τομέα της υγείας, ενώ η ανάγκη για ισχυρά και αξιόπιστα προγνωστικά μοντέλα είναι ακόμη πιο επιτακτική στις ΜΕΘ, όπου νοσηλεύονται ασθενείς σε κρίσιμη κατάσταση. Εμείς ορίζουμε το συγκεκριμένο πρόβλημα ως ένα πρόβλημα ταξινόμησης σε δύο κλάσεις, όπου η πρώτη περιέχει τους ασθενείς που κατέληξαν κατά τη διάρκεια της παραμονής τους στη ΜΕΘ (θετική κλάση) και η δεύτερη εκείνους που πήραν εξιτήριο (αρνητική κλάση). Τα δεδομένα εκπαίδευσης είναι οι μετρήσεις ζωτικών ενδείξεων και τα εργαστηρια-

κά αποτελέσματα των ασθενών, με τη μορφή πολυμεταβλητών χρονοσειρών, από το πρώτο 24ωρο της παραμονής τους στη ΜΕΘ, ενώ επικεντρωνόμαστε στην πρόβλεψη του κινδύνου θνησιμότητας για το επόμενο 48ωρο, δηλαδή τη 2η και 3η ήμερα της παραμονής στη ΜΕΘ.

Ο στόχος της εργασίας είναι η αξιολόγηση της επίδοσης διαφορετικών αλγορίθμων ομοσπονδιακής μάθησης, συγκριτικά με την ιδανική, αλλά μη ασφαλή, κεντρική μηχανική μάθηση και με την ασφαλή, αλλά λιγότερο αποδοτική, τοπική μηχανική μάθηση, για το παραπάνω πρόβλημα. Το ομοσπονδιακό σύνολο νοσοκομείων αποτελείται από πραγματικά δεδομένα, από την συνεργατική, ερευνητική βάση δεδομένων eICU [17], δημιουργώντας έτσι ένα ρεαλιστικό, ετερογενές πειραματικό περιβάλλον. Αρχικά, ερευνούμε πόσο καλά ενσωματώνονται διαφορετικά βαθιά αναδρομικά νευρωνικά δίκτυα στο πλαίσιο της ομοσπονδιακής μάθησης, αναφορικά με το συγκεκριμένο πρόβλημα, καθώς και την ικανότητα γενίκευσης των ομοσπονδιακών μοντέλων πάνω σε ένα testing σύνολο με δεδομένα από νοσοκομεία που δεν συμμετείχαν στη φάση της εκπαίδευσης, συγκριτικά με την κεντρική και τοπική προσέγγιση. Έπειτα, ελέγχουμε την ευαισθησία των αλγορίθμων ομοσπονδιακής μηχανικής μάθησης σε ένα ομοσπονδιακό περιβάλλον που περιλαμβάνει τεχνητά νοσοκομεία με ακραία χαρακτηριστικά, σε σχέση με το μέγεθος και την κατανομή των δεδομένων τους. Τέλος, εξετάζουμε την επίδραση της συμμετοχής μεμονωμένων νοσοκομείων σε ένα ομοσπονδιακό περιβάλλον εκπαίδευσης, αξιολογώντας την επίδοση των ομοσπονδιακών μοντέλων πάνω στα τοπικά τους δεδομένα.

0.2 Θεωρητικό Υπόβαθρο

Σε αυτήν την ενότητα παρουσιάζουμε κάποιες βασικές θεωρητικές έννοιες, οι οποίες είναι χρήσιμες για την καλύτερη κατανόηση του αντικειμένου της διπλωματικής εργασίας, καθώς και σχετικές ερευνητικές εργασίες.

0.2.1 Χρονοσειρές

Η ευρεία χρήση συσκευών παρακολούθησης της υγείας των ασθενών και η υιοθέτηση των ηλεκτρονικών φακέλων υγείας έχουν οδηγήσει στη ραγδαία αύξηση των διαθέσιμων ιατρικών δεδομένων, με τη μορφή χρονοσειρών. Οι *χρονοσειρές* αποτελούν μια συλλογή από παρατηρήσεις ταξινομημένες σε χρονολογική σειρά. Αναφερόμαστε στο σύνολο των μεθόδων για την εξαγωγή στατιστικών στοιχείων και χαρακτηριστικών των χρονολογικών σειρών ως *ανάλυση χρονοσειρών*, ενώ η χρήση μοντέλων για την πρόβλεψη μελλοντικών τιμών μιας χρονοσειράς, σύμφωνα με τις προηγούμενες τιμές, ονομάζεται *πρόβλεψη χρονοσειρών* [18]. Ανάλογα με το πλήθος των καταγεγραμμένων χαρακτηριστικών ή μεταβλητών, οι χρονοσειρές διακρίνονται σε μονομεταβλητές και πολυμεταβλητές, όπως αυτές που απεικονίζονται στην Εικόνα 1.

Για την βαθύτερη κατανόηση των χρονοσειρών, είναι σημαντικό να μελετηθούν ορισμένα βασικά χαρακτηριστικά τους. Η τάση, η εποχικότητα, η κυκλικότητα και η στασιμότητα είναι οι κυριότερες συνιστώσες που απαρτίζουν μια χρονοσειρά και μπορούν να αποδομηθούν και να μελετηθούν μεμονωμένα μέσω τεχνικών αποσύνθεσης χρονοσειρών. Συχνά στις χρονοσειρές παρατηρούνται *ακραίες τιμές*, δηλαδή παρατηρήσεις που δεν ακολουθούν τη συμπεριφορά που ορίζεται από τα υπόλοιπα δεδομένα, οι οποίες δεν μπορούν να ερμηνευ-

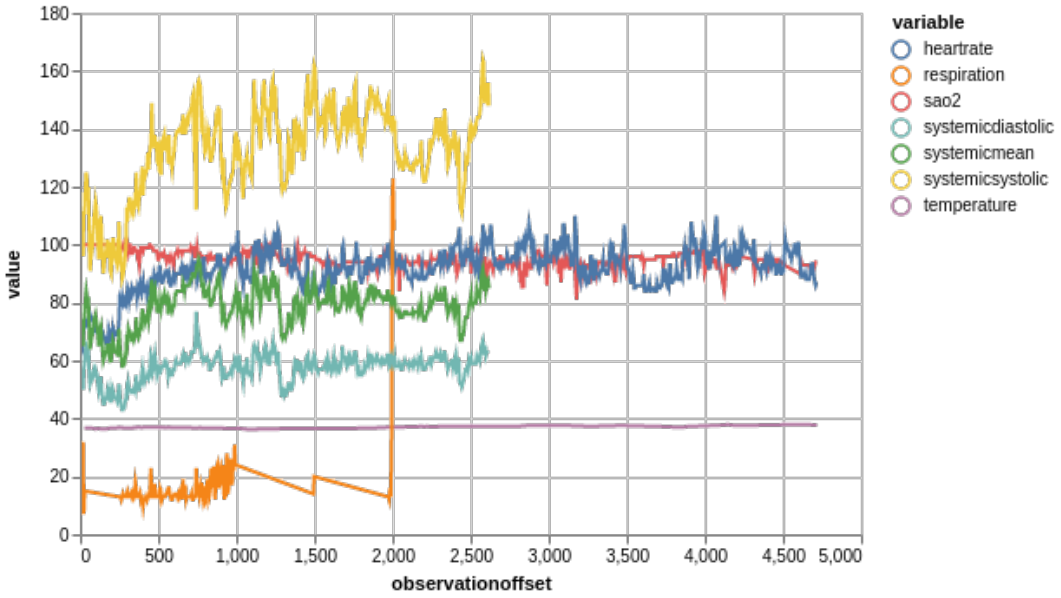


Figure 1. *Μετρήσεις ζωτικών ενδείξεων ενός ασθενή κατά τις πρώτες ώρες της παραμονής του στη ΜΕΘ [1]*

θούν μέσω της αποσύνθεσης και απαιτούν ειδική μεταχείριση. Τέτοιες τιμές μπορούν να εντοπιστούν μέσω στατιστικών μεθόδων και να αντιμετωπιστούν με διαγραφή, αντικατάσταση με τις κοντινότερες μη-ακραίες παρατηρήσεις [19] ή με τη χρήση τεχνικών ανάλυσης που είναι ανθεκτικές στην ύπαρξη ακραίων τιμών [20].

Έπειτα, οι πολυμεταβλητές χρονολογικές σειρές ενδέχεται να περιλαμβάνουν αλληλεξαρτήσεις ανάμεσα στις μεταβλητές, οι οποίες μπορεί να οδηγήσουν σε λανθασμένα συμπεράσματα αν δεν εντοπιστούν και ληφθούν υπόψη. Ένα ακόμα χαρακτηριστικό των πολυμεταβλητών χρονοσειρών είναι ο υψηλός αριθμός διαστάσεων, ο οποίος αυξάνει την πολυπλοκότητα και δυσκολεύει την ανάλυση και οπτικοποίηση των δεδομένων. Μία συνηθισμένη διαδικασία αντιμετώπισης αυτών των προκλήσεων είναι η *εξαγωγή χαρακτηριστικών* κατά την προεπεξεργασία των χρονοσειρών. Αρχικά, μπορεί να επιλεγεί ένα υποσύνολο των διαθέσιμων μεταβλητών, διατηρώντας τις πιο σημαντικές πληροφορίες προς επίλυση του εξεταζόμενου προβλήματος και μειώνοντας τις διαστάσεις της χρονοσειράς [21]. Έπειτα, οι αρχικές μεταβλητές μπορούν να τροποποιηθούν και να συνδυαστούν μέσω κατάλληλων αλγορίθμων για να παράξουν νέες χρονοσειρές, οι οποίες αντιπροσωπεύουν πιο σύνθετες έννοιες των χρονοσειρών, και να βελτιώσουν τα αποτελέσματα της ανάλυσης και των μοντέλων πρόβλεψης [22]. Στο ίδιο αποτέλεσμα αποσκοπεί και η κανονικοποίηση των δεδομένων, καθώς μεταβλητές εκφρασμένες σε υψηλότερη κλίμακα μπορεί να κυριαρχούν εις βάρος άλλων κατά την εκπαίδευση ενός προγνωστικού μοντέλου.

Τέλος, η ύπαρξη *απουσιαζουσών τιμών* είναι ένα βασικό πρόβλημα των χρονοσειρών και η αποτελεσματική αντιμετώπισή του επηρεάζει άμεσα την επίδοση ενός προγνωστικού μοντέλου. Ένας ενδεχόμενος τρόπος αντιμετώπισης είναι η απόρριψη των χρονοσειρών που εμφανίζουν απουσιάζουσες τιμές, ωστόσο αυτή η προσέγγιση οδηγεί σε σημαντική απώλεια πληροφορίας. Άλλες προσεγγίσεις επικεντρώνονται στην αντικατάσταση των απουσιαζουσών τιμών με τις πιο κοντινές παρατηρήσεις σε αυτές ή με τον μέσο όρο τους [4]. Πιο σύνθετες

τεχνικές υπολογίζουν μια εκτίμηση της απουσιάζουσας τιμής εφαρμόζοντας μια γραμμική ή πολυωνυμική συνάρτηση στις παρατηρήσεις γύρω της, οι οποίες είναι προτιμότερες όταν οι απουσιάζουσες τιμές είναι σποραδικές.

0.2.2 Βαθιά Νευρωνικά Δίκτυα

Η *μηχανική μάθηση* είναι ένας υποκλάδος της Τεχνητής Νοημοσύνης που διερευνά τη μελέτη και κατασκευή μοντέλων που δίνουν στους υπολογιστές την ικανότητα να ¨μαθαίνουν¨, δηλαδή να αξιοποιούν αποτελεσματικά ένα σύνολο δεδομένων για να βελτιώσουν την επίδοσή τους πάνω σε ένα σετ εργασιών [23]. Οι τρεις βασικές κατηγορίες της μηχανικής μάθησης είναι η επιβλεπόμενη, η μη επιβλεπόμενη και η ενισχυτική μάθηση. Το πρόβλημα αυτής της διπλωματικής εργασίας ανήκει στην πρώτη κατηγορία, καθώς κάθε σύνολο εισόδου, δηλαδή οι ζωτικές ενδείξεις και τα εργαστηριακά αποτελέσματα ενός ασθενή, συνδέεται με μια επιθυμητή τιμή εξόδου, την κατάληξη ή το εξιτήριο του ασθενή από τη ΜΕΘ.

Τα *τεχνητά νευρωνικά δίκτυα* αποτελούν δίκτυα απλών υπολογιστικών κόμβων, εμπνευσμένα από τη διασύνδεση των βιολογικών νευρώνων του ανθρώπινου εγκεφάλου. Κάθε τέτοιος υπολογιστικός κόμβος (ή νευρώνας) δέχεται ένα σύνολο αριθμητικών εισόδων από το προηγούμενο επίπεδο, επιτελεί έναν υπολογισμό πάνω σε αυτό και προωθεί το αποτέλεσμα στους νευρώνες του επόμενου επιπέδου ή στην έξοδο, πολλαπλασιασμένο με κάποιο συναπτικό βάρος που αντιστοιχεί στη συγκεκριμένη σύνδεση [24]. Κατά της διάρκειας της εκπαίδευσης ενός νευρωνικού μοντέλου με επιβλεπόμενη μάθηση, το μοντέλο επεξεργάζεται με την παραπάνω διαδικασία την πληροφορία από το σύνολο εισόδου για να παράξει μία έξοδο [2]. Αυτή η εκτιμώμενη έξοδος συγκρίνεται με την επιθυμητή έξοδο και το νευρωνικό μοντέλο προσαρμόζει τα βάρη του δικτύου ανάλογα με το μέγεθος του σφάλματος. Η ιδέα των νευρωνικών δικτύων αποτέλεσε το ερέθισμα για τη *βαθιά μάθηση*, μια τεχνική μηχανικής μάθησης που τοποθετεί πολλά επίπεδα νευρώνων σε ένα μοντέλο, δημιουργώντας έτσι υψηλού επιπέδου αναπαραστάσεις των δεδομένων εισόδου [25].

Τα νευρωνικά δίκτυα χωρίζονται σε δύο βασικές κατηγορίες: στα νευρωνικά δίκτυα *πρόσθιας τροφοδότησης*, που δεν περιλαμβάνουν καμία αναδρομική σύνδεση, και στα *αναδρομικά* νευρωνικά δίκτυα, με τουλάχιστον ένα βρόγχο ανάδρασης. Σύμφωνα με τη βασική αρχιτεκτονική αναδρομικών νευρωνικών δικτύων, οι εσωτερικοί κόμβοι ενός επιπέδου λαμβάνουν την έξοδο του προηγούμενου επιπέδου συγχρονισμένα , καθώς και την τιμή του γειτονικού τους κόμβου για τα δεδομένα της προηγούμενης χρονικής στιγμής δυναμικά, για να υπολογίσουν την τιμή τους [2]. Ο τρόπος με τον οποίο επεξεργάζονται την πληροφορία και το γεγονός ότι μπορούν να επεξεργαστούν εισόδους μεταβλητού μεγέθους [26] καθιστούν τα αναδρομικά δίκτυα ιδανικά για εφαρμογές στα πεδία της επεξεργασίας φωνής και φυσικής γλώσσας, της πρόβλεψης χρονοσειρών και, γενικότερα, σε εργασίες που αφορούν σειριακά δεδομένα. Τα αναδρομικά νευρωνικά δίκτυα δίνουν την αίσθηση του βάθους στη διάσταση του χρόνου, ωστόσο, όπως φαίνεται στο Σχήμα 2, μπορούν επίσης να οδηγήσουν σε μοντέλα βαθιάς μάθησης με την τοποθέτηση πολλών εσωτερικών επιπέδων στη σειρά.

Εκτός της βασικής αρχιτεκτονικής ενός αναδρομικού νευρωνικού δικτύου, έχουν δημιουργηθεί και πολλές παραλλαγές της. Το μοντέλο Long Short-Term Memory (LSTM) προσθέτει μια σειρά μηχανισμών σε κάθε εσωτερικό κόμβο, οι οποίοι ρυθμίζουν το μέγεθος

της επίδρασης της εισόδου στην τιμή του κόμβου και του δίνουν την ικανότητα διατηρεί πληροφορίες από δεδομένα προηγούμενων χρονικών στιγμών, μια ένδειξη ¨μακράς μνήμης¨ [2]. Το μοντέλο Gated Recurrent Unit (GRU) ακολουθεί την προσέγγιση του LSTM, ενώ μειώνει το πλήθος των απαιτούμενων μηχανισμών κάθε κόμβου, οδηγώντας σε ταχύτερους υπολογισμούς και χαμηλότερες υπολογιστικές απαιτήσεις [2] (Σχήμα 3). Τέλος, μια ακόμα προσέγγιση είναι η χρήση αμφίδρομων αναδρομικών δικτύων, που επεξεργάζονται τα σειριακά δεδομένα εισόδου ως προς την πραγματική και την ανάποδη κατεύθυνση, καθιστώντας τα ιδανικά για εφαρμογές όπου οι μελλοντικές τιμές μιας χρονοσειράς ενέχουν σημαντικές πληροφορίες για τον προσδιορισμό των προηγούμενων τιμών [2].

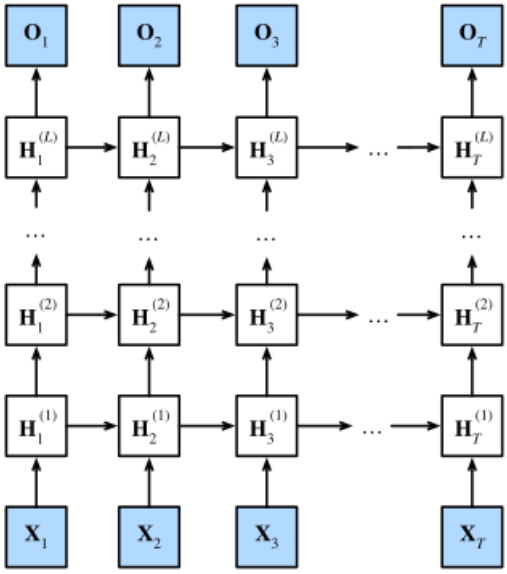


Figure 2. *Αναδρομικό Νευρωνικό Δίκτυο Πολλών Επιπέδων [2]*

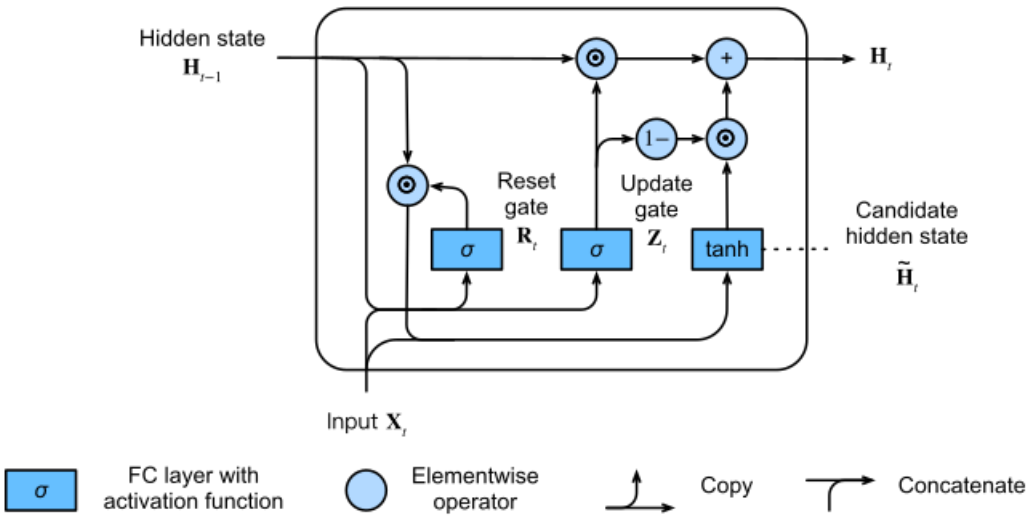


Figure 3. *Η αρχιτεκτονική ενός κόμβου GRU [2]*

0.2.3 Ομοσπονδιακή Μάθηση

Η *ομοσπουδιακή μάθηση* είναι μια τεχνική μηχανικής μάθησης, όπου ένας κεντρικός εξυπηρετητής συντονίζει την εκπαίδευση ενός μοντέλου πάνω σε μια συγκεκριμένη εργασία μάθησης από ένα σύνολο συνεργαζόμενων φορέων, χωρίς τη συγκέντρωση των δεδομένων εκπαίδευσης [27]. Αυτή η προσέγγιση αντιμετωπίζει τις υπολογιστικές δυσκολίες και, σε μεγάλο βαθμό, τα θέματα ασφάλειας και ιδιωτικότητας των συνηθισμένων μεθόδων μηχανικής μάθησης, ενώ, σε αντίθεση με την κατανεμημένη μάθηση, δεν προϋποθέτει πως τα τοπικά σύνολα δεδομένων είναι ανεξάρτητα, ομοιόμορφα κατανεμημένα και με παρεμφερή μεγέθη. Συγχρόνως, ένα περιβάλλον ομοσπονδιακής μάθησης μπορεί να περιλαμβάνει μεγάλο πλήθος συνεργαζόμενων συσκευών (τυπικά μεγαλύτερο από το κατά μέσο όρο μέγεθος των τοπικών συνόλων δεδομένων), καθώς επίσης είναι πιο ανθεκτικό σε αργές ή μη-ανταποκρινόμενες συσκευές χωρίς να αναστέλλει τη διαδικασία εκπαίδευσης.

Ο στόχος της ομοσπονδιακής μάθησης είναι να βρεθεί ένα σύνολο ενημερώσεων των βαρών ενός ¨παγκόσμιου¨ μοντέλου από τους συνεργαζόμενους φορείς, ούτως ώστε να βελτιστοποιηθούν οι παράμετροί του. Προτού ξεκινήσει η διαδικασία της μάθησης, ο κεντρικός εξυπηρετητής επιλέγει και αρχικοποιεί το μοντέλο μηχανικής μάθησης που πρόκειται να εκπαιδευτεί. Ακόμα, ενημερώνει τις συμμετέχουσες συσκευές σχετικά με τη μορφή των δεδομένων που απαιτούνται για την εκπαίδευση [28]. Κάθε γύρος εκπαίδευσης της ομοσπονδιακής μάθησης αποτελείται από τα παρακάτω βήματα:

1. **Επιλογή Συμμετεχόντων:** Ο κεντρικός εξυπηρετητής διαλέγει ένα υποσύνολο των διαθέσιμων φορέων που θα συμμετέχουν σε αυτόν το γύρο εκπαίδευσης [28].

2. **Μετάδοση Πληροφοριών:** Ο κεντρικός εξυπηρετητής αποστέλλει στους επιλεγμένους φορείς τα πιο πρόσφατα βάρη του ¨παγκόσμιου¨ μοντέλου και τη μέθοδο εκπαίδευσης [28].

3. **Εκπαίδευση:** Κάθε επιλεγμένος φορέας ενημερώνει τα βάρη ενός αντίγραφου του ¨παγκόσμιου¨ μοντέλου με εκπαίδευση πάνω στα τοπικά του δεδομένα, ακολουθώντας την προκαθορισμένη μέθοδο εκπαίδευσης [28].

4. **Συγχώνευση Ενημερώσεων:** Οι επιλεγμένοι φορείς αποστέλλουν τα ενημερωμένα τοπικά βάρη τους στον κεντρικό εξυπηρετητή, ο οποίος τα επεξεργάζεται και τα συγχωνεύει με κατάλληλο τρόπο [28].

5. **Ενημέρωση Μοντέλου:** Το ¨παγκόσμιο¨ μοντέλο, που διατηρείται στον κεντρικό εξυπηρετητή, ενημερώνεται σύμφωνα με τους τελευταίους υπολογισμούς. Είτε η εκπαίδευση ολοκληρώνεται και το μοντέλο στέλνεται σε όλους τους συνεργαζόμενους φορείς, είτε επιλέγεται ένα νέο υποσύνολο φορέων για τον επόμενο γύρο ομοσπονδιακής εκπαίδευσης [28].

Στη βαθιά μάθηση, ένας αλγόριθμος με κυρίαρχο ρόλο είναι ο gradient descent. Αυτός ο επαναληπτικός αλγόριθμος βελτιστοποίησης αποσκοπεί στην εύρεση ενός τοπικού ελαχίστου μιας διαφοροποιήσιμης συνάρτησης σφάλματος προσαρμόζοντας τις παραμέτρους

του μοντέλου. Ωστόσο, επειδή η χρήση αυτού του αλγορίθμου επάνω σε ολόκληρο το σύνολο εκπαίδευσης παρουσιάζει υψηλό υπολογιστικό κόστος, ο *στοχαστικός gradient descent* επιλέγει ένα υποσύνολο των δεδομένων εκπαίδευσης και εφαρμόζει τον αλγόριθμο σε αυτά. Ο βασικός αλγόριθμος ομοσπονδιακής μάθησης αποτελεί μια προσαρμογή του στοχαστικού gradient descent στο πλαίσιο της ομοσπονδιακής μάθησης και ονομάζεται FedSGD [29]. Αντί για τη συμμετοχή όλων των συνεργαζόμενων φορέων σε ένα γύρο εκπαίδευσης, αυτός ο αλγόριθμος προτείνει την επιλογή ενός υποσυνόλου αυτών και την εύρεση του μέσου όρου των gradients που υπολογίζουν για την ενημέρωση του ¨παγκόσμιου¨ μοντέλου.

McMahan et al. [27] πρότειναν μια γενίκευση του FedSGD, όπου ο κεντρικός εξυπηρετητής υπολογίζει το μέσο όρο των υπολογιζόμενων τοπικών βαρών, καθώς αυτό ισοδυναμεί με τον μέσο όρο των gradients εφόσον τα τοπικά μοντέλα ξεκινάνε το γύρο εκπαίδευσης με τα ίδια βάρη. Αυτός ο αλγόριθμος ονομάζεται FedAvg και είναι ο πιο ευρέως διαδεδομένος αλγόριθμος ομοσπονδιακής μάθησης. Ωστόσο, αυτός ο αλγόριθμος δεν εγγυάται υψηλή επίδοση και γρήγορη σύγκλιση σε σενάρια ομοσπονδιακής μάθησης, όπου τα τοπικά σύνολα δεδομένων διέπονται από υψηλή στατιστική ετερογένεια. Προς αυτήν την κατεύθυνση, έχουν προταθεί αλγόριθμοι που ρυθμίζουν την εκπαίδευση στο επίπεδο των συνεργαζόμενων φορέων, όπως ο FedProx με έναν παράγοντα εγγύτητας που περιορίζει τις τοπικές ενημερώσεις βαρών πλησιέστερα στο τελευταίο ¨παγκόσμιο¨ μοντέλο [30]. Έπειτα, άλλοι αλγόριθμοι επικεντρώνονται στην βελτιστοποίηση στο επίπεδο του κεντρικού εξυπηρετητή, όπως οι αλγόριθμοι FedAdam, FedAdagrad και FedYogi [31] που προσαρμόζουν τους αντίστοιχους βελτιστοποιητές στην ενημέρωση των βαρών του ¨παγκόσμιου¨ μοντέλου ή ο FedAvgM [32, 33] που προσθέτει έναν παράγοντα momentum στις προηγούμενες ενημερώσεις. Τέλος, αξίζει να σημειωθεί πως στο πλαίσιο της ομοσπονδιακής μάθησης μπορούν να ενσωματωθούν μηχανισμοί ιδιωτικότητας, όπως differential privacy [11], καθώς και μηχανισμοί ασφάλειας, όπως secure aggregation [11].

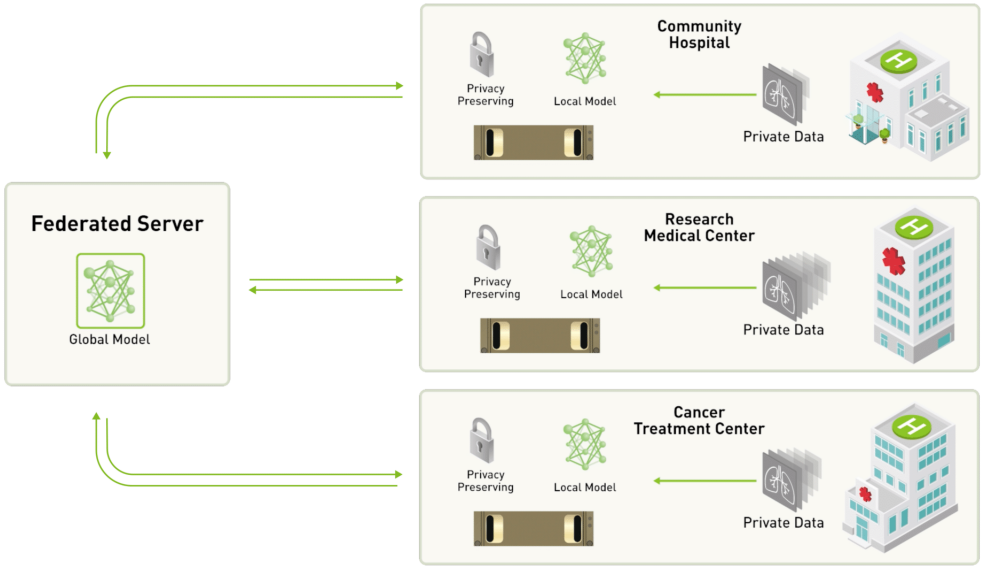


Figure 4. *Ένα σύστημα ομοσπονδιακής μάθησης στον τομέα της υγείας, με έναν κεντρικό ομοσπονδιακό εξυπηρετητή [3]*

0.2.4 Σχετικές Ερευνητικές Εργασίες

Η εκτίμηση του κινδύνου θνησιμότητας ενός ασθενή είναι ένα από τα πιο σημαντικά προγνωστικά προβλήματα στον χώρο της υγείας, με ακόμα μεγαλύτερο αντίκτυπο για περιπτώσεις ασθενών σε μονάδες εντατικής θεραπείας. Για αρκετό καιρό, ο κίνδυνος θνησιμότητας ενός ασθενή υπολογιζόταν με τη χρήση συμβατικών συστημάτων έγκαιρης προειδοποίησης και αξιολόγησης της κατάστασης των ασθενών, όπως το APACHE [34] και το SAPS [35], ωστόσο η ραγδαία αύξηση των χρονολογικών σειριακών δεδομένων σχετικά με την υγεία των ασθενών, σε ηλεκτρονική μορφή, έχει δημιουργήσει την ανάγκη για πιο ισχυρά προγνωστικά συστήματα. Τα τελευταία χρόνια, έχουν διερευνηθεί πολλά διαφορετικά μοντέλα μηχανικής μάθησης και διαφορετικά σύνολα ιατρικών δεδομένων για την αντιμετώπιση αυτού του προβλήματος, οδηγώντας σε αποτελεσματικά προγνωστικά μοντέλα που σημειώνουν αξιοσημείωτες επιδόσεις σε σχέση με τα συμβατικά συστήματα.

Συχνά τα σύνολα τοπικών δεδομένων των κέντρων υγείας είναι περιορισμένα σε μέγεθος και εκτεθειμένα σε ασθενείς με συγκεκριμένα χαρακτηριστικά, γεγονός που έχει οδηγήσει στην έρευνα μεθόδων ομοσπονδιακής μάθησης για τη συνεργατική εκπαίδευση καθολικών προγνωστικών μοντέλων. Πρόσφατα, Dang et al. [14] αξιολόγησαν διάφορους αλγόριθμους ομοσπονδιακής μάθησης στην αιχμή της τεχνολογίας με τη βάση δεδομένων eICU, εκτιμώντας τον κίνδυνο θνησιμότητας κατά την παραμονή των ασθενών στο νοσοκομείο με τη χρήση δημογραφικών και στατιστικών στοιχείων από το πρώτο 24ωρο της παραμονής τους. Έδειξαν πως τα μοντέλα ομοσπονδιακής μάθησης είχαν παρόμοια επίδοση με το ιδανικό μοντέλο κεντρικής μηχανικής μάθησης, ενώ ανέδειξαν τη χρήση χρονολογικών σειριακών δεδομένων και σύνθετων αρχιτεκτονικών μηχανικής μάθησης στο πλαίσιο της ομοσπονδιακής μάθησης ως μελλοντικές ερευνητικές κατευθύνσεις.

Επικεντρώνοντας την έρευνα τους στο χώρο της ΜΕΘ με τη βάση δεδομένων MIMIC-III [36], Mondrejevski et al. [37] πρότειναν το FLICU, μια ροή εργασίας για την εκτίμηση του κινδύνου θνησιμότητας στη ΜΕΘ με χρήση πολυμεταβλητών χρονοσειρών και νευρωνικών δικτύων βαθιάς μάθησης στο πλαίσιο της ομοσπονδιακής μάθησης. Η προσέγγισή τους παρουσίαζε ορισμένους περιορισμούς, όπως το ότι χρησιμοποιούν δεδομένα από τις τελευταίες ώρες της παραμονής των ασθενών στη ΜΕΘ, που αποτρέπουν την εφαρμογή του FLICU για υποστήριξη κλινικών αποφάσεων. Randl et al. [15] προσάρμοσαν αυτή την προτεινόμενη ροή εργασίας στις πρώτες ώρες της παραμονής των ασθενών στη ΜΕΘ, παρατηρώντας πως τα μοντέλα ομοσπονδιακής μάθησης διατηρούν υψηλή επίδοση, κοντά σε εκείνη των ιδανικών μοντέλων κεντρικής μηχανικής μάθησης και σημαντικά καλύτερη από εκείνη των τοπικών μοντέλων, καθώς αυξάνεται το πλήθος των συνεργαζόμενων τεχνητών "ιατρικών κέντρων". Όμως, τα πειράματά τους επίσης παρουσιάζουν κάποια μειονεκτήματα, όπως το γεγονός ότι τα δεδομένα τους πηγάζουν από ένα ιατρικό κέντρο και χωρίζονται στους τεχνητούς φορείς ισοδύναμα και ομοιόμορφα, δημιουργώντας ένα πειραματικό σενάριο που δεν ανταποκρίνεται στον πραγματικό κόσμο. Σε αυτήν τη διπλωματική εργασία, κατασκευάζουμε ένα περιβάλλον ομοσπονδιακής μάθησης με πραγματικά δεδομένα από διαφορετικά νοσοκομεία στις ΗΠΑ, εξετάζοντας πλήθος αλγορίθμων ομοσπονδιακής μάθησης και αναδρομικών νευρωνικών δικτύων υπό ρεαλιστικές συνθήκες στατιστικής ετερογένειας.

0.3 Πειραματικό Μέρος

Σε αυτήν την ενότητα παρουσιάζουμε τη βάση δεδομένων που χρησιμοποιήσαμε, το σχεδιασμό των πειραμάτων και της διαδικασίας εκπαίδευσης των μοντέλων, καθώς και τα αποτελέσματα των πειραμάτων και τα συμπεράσματα που εξάγουμε από αυτά.

0.3.1 Δεδομένα & Προεπεξεργασία

Τα δεδομένα για το πειραματικό μέρος αυτής της εργασίας προέρχονται από τη Συνεργατική Ερευνητική Βάση Δεδομένων eICU [17], την οποία διαχειρίζεται το Εργαστήριο Υπολογιστικής Φυσιολογίας του MIT, και η πρόσβαση σε αυτήν εγκρίνεται έπειτα από την επιτυχημένη ολοκλήρωση ενός προγράμματος σχετικά με ηθικούς και ρυθμιστικούς παράγοντες της έρευνας με πραγματικά δεδομένα ευαίσθητης φύσης. Τα δεδομένα της eICU είναι πραγματικά στοιχεία από περισσότερες από 200 χιλιάδες επισκέψεις ασθενών στη ΜΕΘ σε 208 διαφορετικά νοσοκομεία των ΗΠΑ την περίοδο 2014-2015. Αυτά τα στοιχεία περιλαμβάνουν μετρήσεις ζωτικών ενδείξεων, εργαστηριακές εξετάσεις, φαρμακευτική αγωγή, πλάνο ιατρικής περίθαλψης, διάγνωση, ιστορικό του ασθενή, κ.α. Για την παρούσα εργασία, αναφορικά με το πρόβλημα που εξετάζουμε, χρειαζόμαστε τους πίνακες *patient, vitalperiodic και lab* από τη βάση δεδομένων, οι οποίοι περιέχουν βασικές πληροφορίες για την επίσκεψη στη ΜΕΘ και δημογραφικά στοιχεία των ασθενών, μετρήσεις ζωτικών ενδείξεων και εργαστηριακές εξετάσεις αντίστοιχα.

Καθώς εξετάζουμε το πρόβλημα της εκτίμησης του κινδύνου θνησιμότητας μέσα σε ένα συγκεκριμένο πλαίσιο, με ασθενείς που κατέληξαν ή πήραν εξιτήριο την δεύτερη ή τρίτη ήμερα της παραμονής τους στη ΜΕΘ και χρησιμοποιώντας τις μετρήσεις ζωτικών ενδείξεων και τις εργαστηριακές εξετάσεις από το πρώτο 24ωρο ως δεδομένα εισόδου, απαιτείται να θέσουμε κάποια κριτήρια επιλογής επισκέψεων στη ΜΕΘ από το σύνολο της βάσης δεδομένων. Τα κριτήρια που εφαρμόζουμε είναι τα ακόλουθα:

1. Επιλέγουμε επισκέψεις στη ΜΕΘ όπου το αποτέλεσμά τους (εξιτήριο ή κατάληξη του ασθενή) έχει καταγραφεί.

2. Επιλέγουμε επισκέψεις στη ΜΕΘ όπου οι ζωτικές ενδείξεις του ασθενή καταγράφονταν για τουλάχιστον 24 ώρες, ξεκινώντας από την πρώτη μέτρηση ζωτικών ενδείξεων μετά την επίσημη εισαγωγή στη ΜΕΘ.

3. Επιλέγουμε επισκέψεις στη ΜΕΘ που διήρκησαν τουλάχιστον 24 ώρες, αλλά όχι περισσότερες από 72 ώρες, από την πρώτη μέτρηση ζωτικών ενδείξεων που σημειώσαμε παραπάνω.

4. Επιλέγουμε την πρώτη επίσκεψη σε ΜΕΘ για κάθε ασθενή. Σε περίπτωση που δεν μπορούμε να διακρίνουμε ποια είναι η προγενέστερη επίσκεψη, απορρίπτουμε όλες τις επισκέψεις σε ΜΕΘ για τον συγκεκριμένο ασθενή.

5. Επιλέγουμε επισκέψεις στη ΜΕΘ όπου διεξάχθηκε τουλάχιστον μια εργαστηριακή εξέταση και το αποτέλεσμά της έχει καταγραφεί στο σύστημα.

Demographics	Total	Survival	Death
Number of Patients	55,147	53,052	2,095
Age (years)	63.72 (17.13)	63.53 (17.16)	68.48 (15.53)
ICU stay (hours)	44 (14.4)	43.96 (14.41)	45.21 (14.04)
Gender			
Male	29,845	28,708	1,137
Female	25,288	24,331	957
Unknown	14	13	1
Ethnicity			
Caucasian	42,548	40,891	1,657
African American	5,837	5,642	195
Hispanic / Latino	2,116	2,054	62
Asian	1,012	966	46
Other / Unknown	3,634	3,499	135

Πίνακας 1. *Δημογραφικά στοιχεία των ασθενών μετά την προεπεξεργασία*

Εφαρμόζοντας τα παραπάνω κριτήρια επιλογής, καταλήγουμε με ένα σύνολο έγκυρων επισκέψεων στη ΜΕΘ για το πρόβλημα που μελετάμε. Ο Πίνακας 1 παρουσιάζει κάποια δημογραφικά στοιχεία για αυτό το σύνολο ασθενών.

Το επόμενο στάδιο αφορά την προετοιμασία των δεδομένων για τη διαδικασία εκπαίδευσης της μηχανικής μάθησης. Αρχικά, απορρίπτουμε κάποιες ακραίες τιμές ζωτικών ενδείξεων και εργαστηριακών αποτελεσμάτων. Αυτές οι τιμές ξεπερνούν ένα προκαθορισμένο διάστημα έγκυρων τιμών από τους διαχειριστές της βάσης eICU και οφείλονται κυρίως σε μηχανικά σφάλματα των συσκευών καταγραφής. Έπειτα, επαναδειγματοληπτούμε τις ζωτικές ενδείξεις ανά μία ώρα και τις εργαστηριακές εξετάσεις ανά 8 ώρες, συγχωνεύοντας τις μετρήσεις του κάθε διαστήματος με τον υπολογισμό του μέσου όρου τους. Με αυτόν τον τρόπο, περιορίζουμε την επίδραση εσφαλμένων μετρήσεων στην εκπαίδευση και μειώνουμε το μέγεθος του συνόλου των δεδομένων, διευκολύνοντας έτσι την εκτέλεση των πειραματικών σεναρίων της εργασίας. Τέλος, οι απουσιάζουσες μετρήσεις στις χρονοσειρές συμπληρώνονται πρώτα με ολίσθηση προηγούμενων τιμών προς τα εμπρός και, έπειτα, με ολίσθηση μελλοντικών τιμών προς τα πίσω. Σε περίπτωση που κάποια ζωτική ένδειξη ή εργαστηριακή εξέταση δεν καταγράφηκε κατά το πρώτο 24ωρο για κάποιον ασθενή στη ΜΕΘ, τότε οι απουσιάζουσες τιμές συμπληρώνονται με -1.

0.3.2 Πειραματικός Σχεδιασμός

Μετά από προσεκτική εξέταση σχετικών ερευνητικών εργασιών, σχεδιάσαμε μια σειρά από πειραματικά σενάρια, με στόχο να μας δώσουν μια πιο ολοκληρωμένη εικόνα για την επίδοση των αλγορίθμων ομοσπονδιακής μάθησης σε περιβάλλον στατιστικής ετερογένειας. Στο *πρώτο πειραματικό σενάριο* εξετάζουμε ένα ομοσπονδιακό δίκτυο 8 νοσοκομείων (Πίνακας 2), το οποίο εκπαιδεύει ένα νευρωνικό μοντέλο που αξιολογείται πάνω σε ένα ¨ξένο¨ σύνολο δεδομένων από 4 νοσοκομεία που δεν συμμετέχουν στο ομοσπονδιακό δίκτυο (Πίνακας 3). Στόχος μας είναι να εκτιμήσουμε το επίπεδο ενσωμάτωσης διαφορετικών αρχιτεκτονικών αναδρομικών νευρωνικών δικτύων (RNN, LSTM, GRU), καθώς και να συγκρίνουμε το επίπεδο γενίκευσης ενός μοντέλου ομοσπονδιακής μάθησης πάνω σε ένα ¨ξένο¨ σύνολο δεδομένων,

Datasets	Total	Survival	Death
Hospital A	1,018	977	41 (4.0%)
Hospital B	1,041	972	69 (6.6%)
Hospital C	1,788	1,714	74 (4.1%)
Hospital D	773	746	27 (3.5%)
Hospital E	1,129	1,088	41 (3.6%)
Hospital F	1,344	1,244	100 (7.4%)
Hospital G	930	878	52 (5.6%)
Hospital H	1,316	1,248	68 (5.2%)
Training Set	9,339	8,867	472 (5.1%)

Πίνακας 2. *Οι κατανομές των τοπικών συνόλων δεδομένων για τα 8 νοσοκομεία του ομοσπονδιακού δικτύου*

συγκριτικά με την κεντρική και την τοπική προσέγγιση.

Datasets	Total	Survival	Death
Hospital I	421	398	23 (5.5%)
Hospital J	477	456	21 (4.4%)
Hospital K	420	397	23 (5.5%)
Hospital L	485	460	25 (5.2%)
Test Set	1,803	1,711	92 (5.1%)

Πίνακας 3. *Οι κατανομές των τοπικών συνόλων δεδομένων για τα 4 νοσοκομεία που απαρτίζουν το testing σύνολο*

Το *δεύτερο πειραματικό σενάριο* προσθέτει δύο τεχνητά κατασκευασμένα νοσοκομεία στο ομοσπονδιακό δίκτυο, των οποίων τα τοπικά σύνολα δεδομένων χαρακτηρίζονται από ακραίες κατανομές. Πιο συγκεκριμένα, το νοσοκομείο X1 περιέχει δεδομένα για 1900 ασθενείς, εκ των οποίων μόνο το 0.5% κατέληξε κατά τη διάρκεια της παραμονής του στη ΜΕΘ, ενώ το νοσοκομείο X2 περιέχει δεδομένα για 300 ασθενείς, εκ των οποίων το 25% κατέληξε στη ΜΕΘ (Πίνακας 4). Αυξάνοντας τη στατιστική ετερογένεια του ομοσπονδιακού δικτύου με την προσθήκη των δύο παραπάνω νοσοκομείων, ο στόχος μας είναι να εξετάσουμε την ευαισθησία των αλγορίθμων ομοσπονδιακής μάθησης σε αυτά τα διαφορετικά σενάρια συνεργαζόμενων νοσοκομείων.

Datasets	Total	Survival	Death
Hospital X1	1,900	1,890	10 (0.5%)
Training Set + X1	11,239	10,757	482 (4.3%)
Hospital X2	300	225	75 (25.0%)
Training Set + X2	9,639	9,092	547 (5.7%)

Πίνακας 4. *Οι κατανομές των τοπικών συνόλων δεδομένων για τα τεχνητά νοσοκομεία X1 και X2*

Τέλος, το *τρίτο σενάριο* επικεντρώνεται σε δύο συγκεκριμένα νοσοκομεία από το αρχικό ομοσπονδιακό δίκτυο των 8 συνεργαζόμενων νοσοκομείων, το C, με το μεγαλύτερο τοπικό σύνολο δεδομένων, και το G, με μικρό πλήθος διαθέσιμων τοπικών δεδομένων (Πίνακας 2). Εκπαιδεύοντας νευρωνικά μοντέλα στο πλαίσιο της ομοσπονδιακής μάθησης με και χωρίς

τη συμμετοχή τους, εξετάζουμε την επίδραση που έχει η συμμετοχή τους στη διαδικασία της ομοσπονδιακής εκπαίδευσης στην επίδοση του μοντέλου πάνω στα τοπικά τους δεδομένα. Μέσω αυτής της πειραματικής διαδικασίας, αποσκοπούμε στο να αποκτήσουμε μια καλύτερη ιδέα για το πιθανό κίνητρο ενός νοσοκομείου να συμμετέχει σε ένα ομοσπονδιακό δίκτυο και το όφελος που ενδέχεται να έχει αναφορικά με τα χαρακτηριστικά του.

Το νευρωνικό μοντέλο που κατασκευάζουμε αποτελείται από δύο παράλληλα κανάλια εισόδου, ένα για τις μετρήσεις ζωτικών ενδείξεων και ένα για τις εργαστηριακές εξετάσεις. Κάθε κανάλι εξ αυτών αποτελείται από 3 επίπεδα αναδρομικής αρχιτεκτονικής (RNN, LSTM, GRU) με 16 κόμβους το καθένα, προσδίδοντας "βάθος" στο μοντέλο μας και οδηγώντας σε υψηλού επιπέδου αναπαραστάσεις των αρχικών δεδομένων. Έπειτα, κάθε κανάλι συνοδεύεται από ένα επίπεδο batch normalization για κανονικοποίηση των αναπαραστάσεων πριν τη συγχώνευση τους στην ίδια διάσταση. Στη συνέχεια, δύο πλήρως συνδεδεμένα επίπεδα νευρώνων, με εξόδους μεγέθους 16 και 16 αντίστοιχα, επεξεργάζονται τα συγχωνευμένα χαρακτηριστικά. Τέλος, μια σιγμοειδής συνάρτηση εφαρμόζεται στην έξοδο του τελευταίου πλήρως συνδεδεμένου επιπέδου για να υπολογισθεί το εκτιμώμενο ρίσκο θνησιμότητας στη ΜΕΘ. Καθώς εξετάζουμε ένα πρόβλημα ταξινόμησης σε δύο κλάσεις, χρησιμοποιούμε επίσης την binary cross-entropy ως συνάρτηση σφάλματος και τον αλγόριθμο βελτιστοποίησης Adam, με αρχικό ρυθμό εκπαίδευσης 0.001 που υποδιπλασιάζεται ανά 5 γύρους.

Σε κάθε προσέγγιση που εξετάζουμε, αρχικοποιούμε τα μοντέλα με τα ίδια τυχαία βάρη για να αποφύγουμε φαινόμενα μεροληψίας στα αποτελέσματά μας. Προτού ξεκινήσει η διαδικασία της εκπαίδευσης, τα δεδομένα εισόδου κανονικοποιούνται και, λόγω της δυσαναλογίας δειγμάτων για τις δύο κλάσεις, απονέμονται class weights σε κάθε δείγμα, ώστε οι ενημερώσεις των δειγμάτων των δύο κλάσεων να έχουν ισοδύναμη επίδραση στο μοντέλο. Τα δεδομένα χωρίζονται σε batches για πιο αποδοτική εκπαίδευση, ενώ εφαρμόζεται ένας μηχανισμός έγκαιρης διακοπής της εκπαίδευσης, ο οποίος παρατηρεί το F1-Score πάνω στο σύνολο δεδομένων επαλήθευσης και επιλέγει το μοντέλο με την καλύτερη επίδοση με περίοδο υπομονής 30 γύρων και μέγιστη περίοδο εκπαίδευσης 100 γύρων. Έπειτα, για την προσέγγιση της ομοσπονδιακής μάθησης, θεωρούμε πως σε κάθε γύρο συμμετέχουν όλοι οι συνεργαζόμενοι φορείς και κάθε φορέας εκπαιδεύει το μοντέλο του μία φορά πάνω στα δεδομένα εκπαίδευσής του. Οι αλγόριθμοι ομοσπονδιακής μάθησης περιλαμβάνουν διάφορες παραμέτρους, οι οποίες επιλέχθηκαν μετά από πειραματικό έλεγχο.

Καθώς το σύνολο δεδομένων του προβλήματος δεν είναι ισορροπημένα κατανεμημένο στις δύο κλάσεις, η ακρίβεια (accuracy) των προβλέψεων δεν μπορεί να χρησιμοποιηθεί ως αξιόπιστη μετρική αξιολόγησης των μοντέλων. Η πρώτη μετρική που χρησιμοποιούμε αφορά την καμπύλη ROC (Receiver Operating Characteristic curve), η οποία σχεδιάζεται υπολογίζοντας τις μετρικές recall και specificity σε διαφορετικά thresholds ταξινόμησης των δειγμάτων στις δύο κλάσεις. Υπολογίζοντας το εμβαδόν που περικλείεται από τους άξονες και την καμπύλη ROC (Area Under Receiver Operating Characteristic curve - AUROC), λαμβάνουμε μια μετρική επίδοσης του μοντέλου που δηλώνει την ικανότητά του να ταξινομεί τα δείγματα σωστά. Έπειτα, με αντίστοιχο τρόπο, υπολογίζοντας τις μετρικές precision και recall σε διαφορετικά thresholds ταξινόμησης σχεδιάζουμε την καμπύλη PRC (Precision-Recall Curve), η οποία εκφράζει το tradeoff που καλείται να γίνει ανάμεσα στις δύο μετρικές. Η μετρική AUPRC (Area Under Precision-Recall Curve) είναι εξαιρετικά χρήσιμη σε προβλήματα όπου

η θετική κλάση έχει μεγαλύτερη σημασία και υπάρχει υψηλή ανομοιομορφία στην κατανομή των δεδομένων στις κλάσεις, καθώς οι precision και recall δεν εξαρτώνται από τις σωστές προβλέψεις της κυριαρχούσας αρνητικής κλάσης [38]. Τέλος, η μετρική F1-Score είναι ο αρμονικός μέσος όρος των precision και recall, οπότε ταιριάζει αντίστοιχα στο πρόβλημα που εξετάζουμε. Συνολικά, επικεντρωνόμαστε περισσότερο στις μετρικές AUPRC και F1-Score λόγω της υψηλής ανισοκατανομής του συνόλου δεδομένων μας.

0.3.3 Αποτελέσματα & Συζήτηση

Στο πρώτο πειραματικό σενάριο, εξετάζουμε το μοντέλο που περιγράψαμε με διαφορετικές αρχιτεκτονικές αναδρομικών νευρωνικών δικτύων σε περιβάλλον ομοσπονδιακής μάθησης. Χρησιμοποιούμε το ομοσπονδιακό δίκτυο των 8 νοσοκομείων του Πίνακα 1 και αξιολογούμε τα μοντέλα στο ¨ξένο¨ σύνολο δεδομένων του Πίνακα 2.

Αρχικά, εξετάζουμε το μοντέλο RNN. Η ιδανική προσέγγιση της κεντρικής μηχανικής μάθησης επιτυγχάνει AUROC 0.836, AUPRC 0.400 και F1-Score 0.473, ενώ η βασική μέθοδος της τοπικής μηχανικής μάθησης επιτυγχάνει AUROC 0.764, AUPRC 0.239 και F1-Score 0.252. Οι αλγόριθμοι ομοσπονδιακής μάθησης πέτυχαν αρκετά καλύτερες επιδόσεις από την τοπική μηχανική μάθηση, αλλά χωρίς να προσεγγίζουν εκείνες της κεντρικής μηχανικής μάθησης (με εξαίρεση τη μετρική AUROC). Ο FedProx σημείωσε την καλύτερη επίδοση για τη μετρική AUROC (0.838), ο FedAdam για τη μετρική AUPRC (0.364) και ο FedAvg για τη μετρική του F1-Score (0.335). Τα αποτελέσματα για αυτό το πειραματικό σενάριο με το μοντέλο RNN φαίνονται στον Πίνακα 5.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.836 ± 0.013	0.400 ± 0.009	0.473 ± 0.029	-
LML	0.764 ± 0.009	0.239 ± 0.009	0.252 ± 0.013	-
FedAvg	0.832 ± 0.037	0.348 ± 0.052	**0.335 ± 0.047**	11.2 ± 4.3
FedProx	**0.838 ± 0.021**	0.360 ± 0.039	0.321 ± 0.035	12.5 ± 3.4
FedAdam	0.837 ± 0.022	**0.364 ± 0.037**	0.303 ± 0.007	9.8 ± 1.2
FedAdagrad	0.815 ± 0.016	0.318 ± 0.016	0.327 ± 0.024	9.3 ± 2.3
FedYogi	0.825 ± 0.021	0.329 ± 0.029	0.325 ± 0.041	9.1 ± 1.1
FedAvgM	0.818 ± 0.023	0.322 ± 0.035	0.314 ± 0.035	8.7 ± 2.2

Πίνακας 5. *Επίδοση του μουτέλου RNN πάνω στο ¨ξένο¨ σύνολο δεδομένων*

Έπειτα, εξετάζουμε το μοντέλο LSTM. Η ιδανική προσέγγιση της κεντρικής μηχανικής μάθησης επιτυγχάνει AUROC 0.894, AUPRC 0.499 και F1-Score 0.489, ενώ η βασική μέθοδος της τοπικής μηχανικής μάθησης επιτυγχάνει AUROC 0.803, AUPRC 0.326 και F1-Score 0.384. Οι αλγόριθμοι ομοσπονδιακής μάθησης πάλι πέτυχαν αρκετά καλύτερες επιδόσεις από την τοπική μηχανική μάθηση, πλησιάζοντας αυτή τη φορά περισσότερο στις μετρικές επιδόσεις της κεντρικής μηχανικής μάθησης. Ο FedAvg σημείωσε την καλύτερη επίδοση για τη μετρική AUROC (0.899), ο FedYogi για τη μετρική AUPRC (0.476) και ο FedProx για τη μετρική του F1-Score (0.438). Παρατηρούμε πως το μοντέλο LSTM πετυχαίνει συνολικά καλύτερες επιδόσεις στο συγκεκριμένο πρόβλημα, ενώ ενσωματώνεται πιο αποδοτικά στο περιβάλλον της ομοσπονδιακής μάθησης. Τα αποτελέσματα για αυτό το πειραματικό σενάριο

με το μοντέλο LSTM φαίνονται στον Πίνακα 6.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.894 ± 0.007	0.499 ± 0.020	0.489 ± 0.023	-
LML	0.803 ± 0.034	0.326 ± 0.038	0.384 ± 0.020	-
FedAvg	**0.899 ± 0.006**	0.446 ± 0.037	0.417 ± 0.035	16.8 ± 3.9
FedProx	0.897 ± 0.010	0.456 ± 0.032	**0.438 ± 0.014**	18.9 ± 3.0
FedAdam	0.895 ± 0.009	0.454 ± 0.029	0.409 ± 0.028	16.5 ± 1.8
FedAdagrad	0.897 ± 0.010	0.449 ± 0.035	0.422 ± 0.036	15.4 ± 1.7
FedYogi	0.893 ± 0.010	**0.476 ± 0.022**	0.424 ± 0.040	14.7 ± 0.8
FedAvgM	0.897 ± 0.011	0.453 ± 0.038	0.418 ± 0.031	17.2 ± 3.8

Πίνακας 6. *Επίδοση του μοντέλου LSTM πάνω στο ¨ξένο¨ σύνολο δεδομένων*

Η τελευταία εκτέλεσή μας για αυτό το πειραματικό σενάριο αφορά το μοντέλο GRU. Η προσέγγιση της κεντρικής μηχανικής μάθησης επιτυγχάνει AUROC 0.895, AUPRC 0.539 και F1-Score 0.541, ενώ η μέθοδος της τοπικής μηχανικής μάθησης, που εκπαιδεύει ένα τοπικό μοντέλο για κάθε νοσοκομείο, επιτυγχάνει AUROC 0.807, AUPRC 0.360 και F1-Score 0.413. Οι αλγόριθμοι ομοσπονδιακής μάθησης ξεπέρασαν ξανά τις επιδόσεις της τοπικής μηχανικής μάθησης, ενώ είχαν μια αισθητή απόσταση από τις επιδόσεις της κεντρικής μηχανικής μάθησης. Ο FedAdagrad σημείωσε την καλύτερη επίδοση για τη μετρική AUROC (0.892), καθώς και για τη μετρική F1-Score (0.512), ενώ ο FedProx πέτυχε την υψηλότερη επίδοση για τη μετρική του AUPRC (0.507). Παρατηρούμε πως το μοντέλο GRU πετυχαίνει τις καλύτερες επιδόσεις σε αυτό το πειραματικό σενάριο για τις μετρικές AUPRC και F1-Score, ενώ επίσης είναι πιο ¨ελαφρύ¨ υπολογιστικά σε σχέση με το μοντέλο LSTM. Ωστόσο, οι επιδόσεις των μοντέλων ομοσπονδιακής μάθησης παρουσιάζουν μεγαλύτερη απόκλιση από εκείνες της κεντρικής μηχανικής μάθησης, σε σχέση με το μοντέλο LSTM. Τα αποτελέσματα για αυτό το πειραματικό σενάριο με το μοντέλο GRU φαίνονται στον Πίνακα 7.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.895 ± 0.002	0.539 ± 0.004	0.541 ± 0.020	-
LML	0.807 ± 0.032	0.360 ± 0.021	0.413 ± 0.024	-
FedAvg	0.890 ± 0.007	0.499 ± 0.020	0.489 ± 0.023	15.2 ± 2.3
FedProx	0.891 ± 0.006	**0.507 ± 0.016**	0.472 ± 0.042	14.2 ± 2.5
FedAdam	0.891 ± 0.005	0.505 ± 0.014	0.480 ± 0.046	16.1 ± 2.1
FedAdagrad	**0.892 ± 0.006**	0.502 ± 0.018	**0.512 ± 0.032**	15.5 ± 1.7
FedYogi	0.887 ± 0.007	0.500 ± 0.021	0.475 ± 0.030	13.7 ± 0.7
FedAvgM	0.889 ± 0.008	0.502 ± 0.020	0.466 ± 0.039	13.9 ± 0.9

Πίνακας 7. *Επίδοση του μοντέλου GRU πάνω στο ¨ξένο¨ σύνολο δεδομένων*

Στο δεύτερο πειραματικό σενάριο, εξετάζουμε το μοντέλο GRU σε ένα ομοσπονδιακό δίκτυο με την προσθήκη τεχνητών νοσοκομείων με ακραίες κατανομές των δεδομένων τους στις δύο κλάσεις, όπως φαίνονται στον Πίνακα 3. Στο πρώτο ομοσπονδιακό δίκτυο με την προσθήκη του νοσοκομείου X1, η προσέγγιση της κεντρικής μηχανικής μάθησης, που εκπαιδεύει ένα 'παγκόσμιο' μοντέλο με πρόσβαση σε όλα τα δεδομένα εκπαίδευσης, επιτυγχάνει AUROC 0.870, AUPRC 0.461 και F1-Score 0.498, ενώ η μέθοδος της τοπικής μηχανικής

μάθησης επιτυγχάνει AUROC 0.689, AUPRC 0.185 και F1-Score 0.289. Οι αλγόριθμοι ομοσπονδιακής μάθησης επηρεάζονται από την πρόσθετη στατιστική ετερογένεια του δικτύου, ωστόσο πάλι ξεπερνώντας τις επιδόσεις της τοπικής μηχανικής μάθησης. Ο FedProx σημείωσε την καλύτερη επίδοση για τη μετρική AUROC (0.854), ο FedAvg για τη μετρική AUPRC (0.405) και ο FedYogi για τη μετρική του F1-Score (0.433). Τα αποτελέσματα για αυτό το πειραματικό σενάριο με την προσθήκη του νοσοκομείου X1 φαίνονται στον Πίνακα 8.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.870 ± 0.004	0.461 ± 0.046	0.498 ± 0.042	-
LML	0.689 ± 0.030	0.185 ± 0.028	0.289 ± 0.046	-
FedAvg	0.852 ± 0.008	**0.405 ± 0.055**	0.407 ± 0.058	15.5 ± 2.3
FedProx	**0.854 ± 0.008**	0.402 ± 0.044	0.413 ± 0.47	16.5 ± 1.2
FedAdam	0.851 ± 0.012	0.392 ± 0.056	0.400 ± 0.072	15.6 ± 2.0
FedAdagrad	0.848 ± 0.007	0.390 ± 0.049	0.426 ± 0.060	17.2 ± 2.4
FedYogi	0.850 ± 0.008	0.399 ± 0.051	**0.433 ± 0.050**	16.1 ± 1.4
FedAvgM	0.851 ± 0.006	0.392 ± 0.042	0.395 ± 0.074	15.7 ± 2.5

Πίνακας 8. *Επίδοση του μοντέλου στο ομοσπονδιακό δίκτυο με το νοσοκομείο X1*

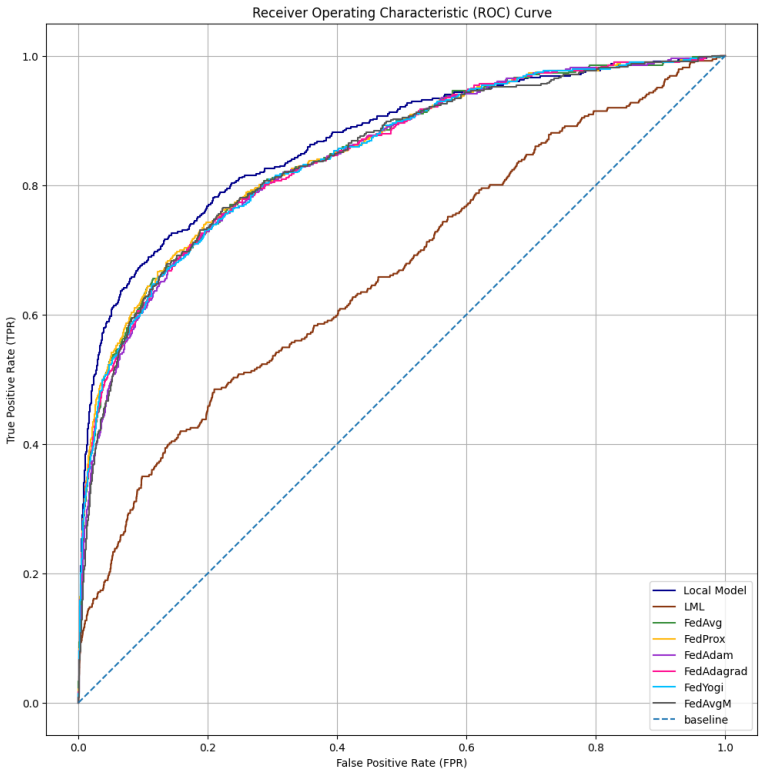


Figure 5. *Σύγκριση των καμπυλών ROC για τα μοντέλα στο ομοσπονδιακό δίκτυο με το νοσοκομείο X1*

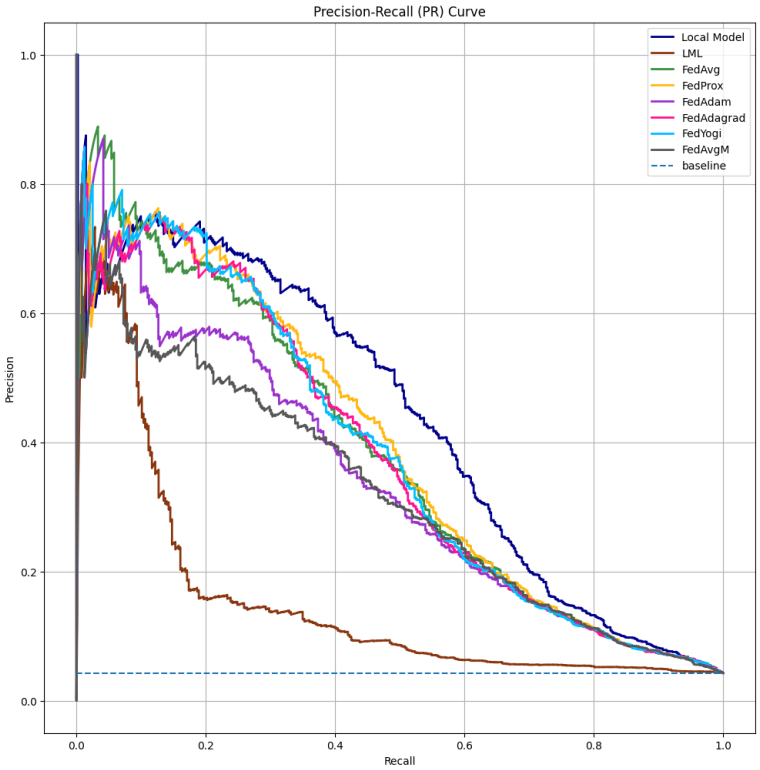


Figure 6. *Σύγκριση των καμπυλών PRC για τα μοντέλα στο ομοσπονδιακό δίκτυο με το νοσοκομείο X1*

Στο δεύτερο ομοσπονδιακό δίκτυο με την προσθήκη του νοσοκομείου X2, το μοντέλο της κεντρικής μηχανικής μάθησης επιτυγχάνει AUROC 0.871, AUPRC 0.537 και F1-Score 0.526, ενώ τα μοντέλα της τοπικής μηχανικής μάθησης επιτυγχάνουν AUROC 0.728, AUPRC 0.288 και F1-Score 0.390. Οι αλγόριθμοι ομοσπονδιακής μάθησης συνολικά αξιοποιούν καλύτερα τις διαθέσιμες πληροφορίες σε αυτό το ομοσπονδιακό δίκτυο. Ο FedAvgM σημείωσε τις καλύτερες επιδόσεις για τις μετρικές AUROC (0.860) και AUPRC (0.503), ενώ ο FedYogi πέτυχε το υψηλότερο σκορ για τη μετρική F1-Score (0.477). Τα αποτελέσματα για αυτό το πειραματικό σενάριο με την προσθήκη του νοσοκομείου X2 φαίνονται στον Πίνακα 9.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.871 ± 0.012	0.537 ± 0.029	0.526 ± 0.032	-
LML	0.728 ± 0.032	0.288 ± 0.037	0.390 ± 0.025	-
FedAvg	0.859 ± 0.011	0.481 ± 0.038	0.447 ± 0.062	19.2 ± 2.0
FedProx	0.859 ± 0.013	0.492 ± 0.027	0.405 ± 0.052	19.6 ± 3.8
FedAdam	0.857 ± 0.013	0.495 ± 0.032	0.452 ± 0.048	19.1 ± 2.6
FedAdagrad	0.857 ± 0.013	0.492 ± 0.034	0.463 ± 0.054	19.3 ± 1.5
FedYogi	0.854 ± 0.012	0.501 ± 0.036	**0.477 ± 0.038**	19.2 ± 3.2
FedAvgM	**0.860 ± 0.011**	**0.503 ± 0.026**	0.463 ± 0.052	20.2 ± 3.2

Πίνακας 9. *Επίδοση του μοντέλου στο ομοσπονδιακό δίκτυο με το νοσοκομείο X2*

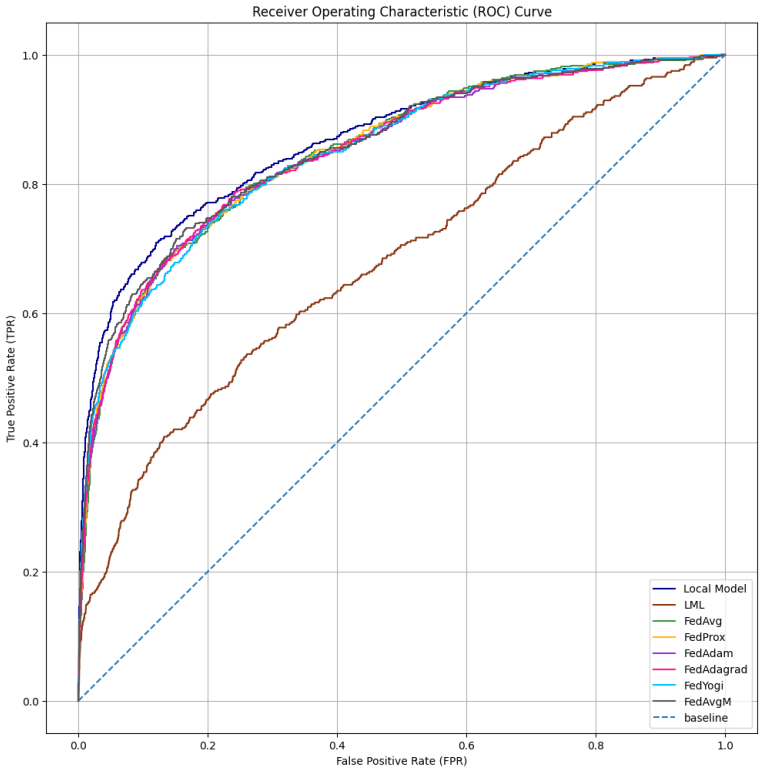


Figure 7. *Σύγκριση των καμπυλών ROC για τα μοντέλα στο ομοσπουδιακό δίκτυο με το νοσοκομείο X2*

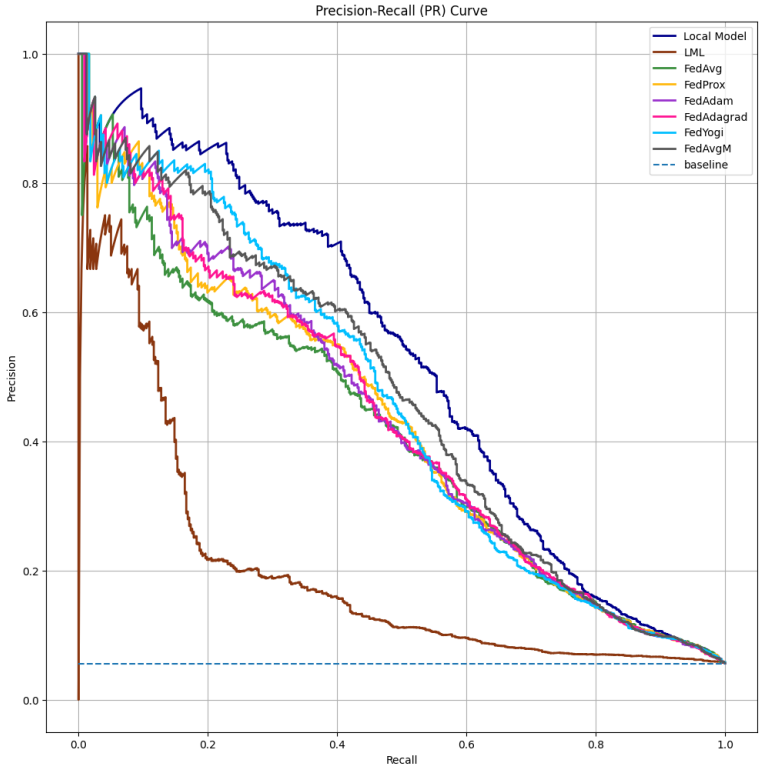


Figure 8. *Σύγκριση των καμπυλών PRC για τα μοντέλα στο ομοσπουδιακό δίκτυο με το νοσοκομείο X2*

Το τρίτο πειραματικό σενάριο επικεντρώνεται σε δύο συγκεκριμένα νοσοκομεία, ώστε να εξετάσει την επίδοση μοντέλων ομοσπονδιακής μάθησης πάνω στα τοπικά τους δεδομένα. Το πρώτο εξ αυτών είναι το νοσοκομείο G με ένα σύνολο τοπικών δεδομένων σχετικά μικρού μεγέθους, όπως φαίνεται στον Πίνακα 1. Το τοπικό μοντέλο του νοσοκομείου G επιτυγχάνει AUROC 0.775, AUPRC 0.322 και F1-Score 0.320, ενώ η προσέγγιση της ομοσπονδιακής μάθησης χωρίς το νοσοκομείο G επιτυγχάνει AUROC 0.827, AUPRC 0.282 και F1-Score 0.341. Οι αλγόριθμοι ομοσπονδιακής μάθησης με το νοσοκομείο G πέτυχαν αρκετά καλύτερες επιδόσεις από τα μοντέλα των προηγούμενων δύο μεθόδων, καθώς εκτέθηκαν κατά τη διάρκεια της εκπαίδευσης και σε δεδομένα του νοσοκομείου G και προσαρμόστηκαν κατάλληλα ως προς τα χαρακτηριστικά του. Ο FedYogi σημείωσε την καλύτερη επίδοση σε όλες τις μετρικές κατηγορίες με AUROC 0.888, AUPRC 0.497 και F1-Score 0.488. Τα αποτελέσματα για αυτό το πειραματικό σενάριο φαίνονται στον Πίνακα 10.

Method	AUROC	AUPRC	F1-Score	Best FL Round
LML	0.775 ± 0.055	0.322 ± 0.156	0.320 ± 0.117	-
TransferFL	0.827 ± 0.011	0.282 ± 0.057	0.341 ± 0.033	17.0 ± 1.2
FedAvg	0.887 ± 0.036	0.492 ± 0.129	0.470 ± 0.138	14.7 ± 1.1
FedProx	0.887 ± 0.037	0.494 ± 0.130	0.444 ± 0.084	15.3 ± 1.3
FedAdam	0.886 ± 0.036	0.488 ± 0.126	0.465 ± 0.095	14.0 ± 0.8
FedAdagrad	0.885 ± 0.038	0.488 ± 0.131	0.466 ± 0.159	14.6 ± 1.1
FedYogi	**0.888 ± 0.039**	**0.497 ± 0.135**	**0.488 ± 0.142**	14.4 ± 1.0
FedAvgM	0.886 ± 0.034	0.492 ± 0.129	0.414 ± 0.073	13.6 ± 1.4

Πίνακας 10. *Επίδοση του μοντέλου στο testing σύνολο του νοσοκομείου G*

Έπειτα, το δεύτερο περιβάλλον που εξετάζουμε σε αυτό το σενάριο επικεντρώνεται στο νοσοκομείο C, που περιλαμβάνει το μεγαλύτερο πλήθος δεδομένων αναφορικά με τα νοσοκομεία του ομοσπονδιακού δικτύου εκπαίδευσης του Πίνακα 1. Το τοπικό μοντέλο του νοσοκομείου C επιτυγχάνει AUROC 0.792, AUPRC 0.324 και F1-Score 0.328, ενώ η προσέγγιση της ομοσπονδιακής μάθησης χωρίς το νοσοκομείο C επιτυγχάνει AUROC 0.802, AUPRC 0.244 και F1-Score 0.320. Οι αλγόριθμοι ομοσπονδιακής μάθησης με το νοσοκομείο C ξανά οδηγούν σε μοντέλα με τις καλύτερες επιδόσεις στις μετρικές κατηγορίες που εξετάζουμε. Ο FedYogi σημειώνει την καλύτερη επίδοση για τη μετρική AUROC (0.814), ο FedAdam για τη μετρική AUPRC (0.382) και ο FedAdagrad για τη μετρική F1-Score (0.388). Τα αποτελέσματα για αυτό το πειραματικό σενάριο φαίνονται στον Πίνακα 11.

Method	AUROC	AUPRC	F1-Score	Best FL Round
LML	0.792 ± 0.056	0.324 ± 0.154	0.328 ± 0.049	-
TransferFL	0.802 ± 0.033	0.244 ± 0.026	0.320 ± 0.026	17.9 ± 1.6
FedAvg	0.811 ± 0.058	0.371 ± 0.124	0.382 ± 0.136	14.8 ± 1.2
FedProx	0.810 ± 0.060	0.373 ± 0.132	0.367 ± 0.142	16.0 ± 1.4
FedAdam	0.812 ± 0.058	**0.382 ± 0.135**	0.347 ± 0.119	15.7 ± 0.7
FedAdagrad	0.811 ± 0.061	0.370 ± 0.140	**0.388 ± 0.132**	17.9 ± 3.6
FedYogi	**0.814 ± 0.060**	0.376 ± 0.140	0.354 ± 0.134	16.9 ± 0.6
FedAvgM	0.812 ± 0.059	0.378 ± 0.134	0.379 ± 0.140	19.3 ± 1.1

Πίνακας 11. *Επίδοση του μοντέλου στο testing σύνολο του νοσοκομείου C*

Από το πρώτο πειραματικό σενάριο, παρατηρούμε πως το μοντέλο RNN πετυχαίνει τις χειρότερες επιδόσεις πάνω στο ¨ξένο¨ σύνολο δεδομένων, έχοντας όμως την μικρότερη επιβάρυνση αναφορικά με την επικοινωνία πάνω από το δίκτυο, καθώς περιλαμβάνει το μικρότερο αριθμό παραμέτρων και την ταχύτερη σύγκλιση στο καλύτερο F1-Score πάνω στα σύνολα δεδομένων επαλήθευσης. Αντίθετα, το μοντέλο LSTM προσδίδει τη μεγαλύτερη επιβάρυνση στο δίκτυο, αλλά τα μοντέλα ομοσπονδιακής μάθησης προσομοιάζουν περισσότερο τις επιδόσεις του μοντέλου κεντρικής μηχανικής μάθησης. Το μοντέλο GRU φαίνεται πως αποτελεί τη "χρυσή τομή" ανάμεσα σε αυτά τα δύο στοιχεία των πειραμάτων. Έπειτα, παρατηρούμε πως ο αλγόριθμος FedAvg πετυχαίνει υψηλές επιδόσεις για τις περισσότερες μετρικές κατηγορίες, ενώ αντίστοιχα αποτελεσματικός είναι και ο αλγόριθμος FedProx. Οι αλγόριθμοι ομοσπονδιακής μάθησης που αποσκοπούν στη βελτιστοποίηση στο επίπεδο του κεντρικού εξυπηρετητή πετυχαίνουν ταχύτερη σύγκλιση σε σχέση με τους δύο προηγούμενους αλγορίθμους, όμως αυτό φαίνεται πως έχει αντίκτυπο στην εκπαίδευση τους, καθώς δεν σημειώνουν επιδόσεις ανάλογες των FedAvg και FedProx.

Το δεύτερο πειραματικό σενάριο μας δίνει μια εικόνα για την επίδοση των αλγορίθμων ομοσπονδιακής μάθησης σε διαφορετικά ομοσπονδιακά δίκτυα. Το νοσοκομείο X1 έχει μεγάλη επίδραση στο δίκτυο λόγω του μεγάλου μεγέθους του συνόλου δεδομένων του, ενώ έχει μηδαμινή εκπροσώπηση της θετικής κλάσης, η οποία καθιστά την εκπαίδευση του μοντέλου στο νοσοκομείο X1 προδιατεθειμένη προς την αρνητική κλάση. Ως αποτέλεσμα, παρατηρούμε μια πτώση της επίδοσης των μοντέλων ομοσπονδιακής μάθησης σε σχέση με εκείνο της κεντρικής μηχανικής μάθησης, ενώ ο αλγόριθμος FedProx φαίνεται να είναι ο καταλληλότερος για αυτό το ομοσπονδιακό δίκτυο, αφού ο παράγοντας εγγύτητας περιορίζει την μεροληψία του νοσοκομείου X1 κατά την εκπαίδευση. Έπειτα, στην περίπτωση του νοσοκομείου X2, έχουμε μικρή επίδραση στο ΄παγκόσμιο΄ μοντέλο αλλά σημαντική εκπροσώπηση της κλάσης που συνολικά υποεκπροσωπείται. Παρατηρούμε πως ο αλγόριθμος FedProx εδώ μειώνει ακόμα περισσότερο τη συνεισφορά του νοσοκομείου X2, ενώ οι αλγόριθμοι βελτιστοποίησης στο επίπεδο του κεντρικού εξυπηρετητή παρουσιάζουν υψηλότερη επίδοση και καλύτερη διαχείριση των ενημερώσεων αυτού του μικρού τεχνητού νοσοκομείου.

Τέλος, το τρίτο πειραματικό σενάριο εξετάζει την επίδοση μοντέλων ομοσπονδιακής μάθησης πάνω στα τοπικά testing σύνολα συγκεκριμένων νοσοκομείων, προσπαθώντας να δώσει μια εικόνα για το κίνητρο νοσοκομείων με διαφορετικά χαρακτηριστικά να συμμετέχουν σε ένα ομοσπονδιακό δίκτυο. Παρατηρούμε πως το νοσοκομείο G επωφελείται περισσότερο από τα μοντέλα ομοσπονδιακής μάθησης που συμμετέχει, αφού δεν έχει αρκετά δεδομένα για να εκπαιδεύσει ένα ισχυρό τοπικό μοντέλο και το μοντέλο ομοσπονδιακής μάθησης χωρίς τη συμμετοχή του δεν προσαρμόζεται επαρκώς στα δεδομένα του. Έπειτα, το νοσοκομείο C κατέχει ένα μεγαλύτερο σύνολο δεδομένων, που του επιτρέπει να εκπαιδεύσει ένα πιο ισχυρό τοπικό μοντέλο. Το μοντέλο ομοσπονδιακής μάθησης χωρίς τη συμμετοχή του έχει παρόμοια επίδοση με το τοπικό μοντέλο, ενώ τα μοντέλα ομοσπονδιακής μάθησης με τη συμμετοχή του πετυχαίνουν καλύτερη επίδοση, αν και όχι στον βαθμό που βελτιώθηκαν οι μετρικές επίδοσης για το νοσοκομείο G. Το συμπέρασμα από αυτό το πειραματικό σενάριο είναι πώς, ανεξαρτήτως μεγέθους, ένα νοσοκομείο παρακινείται να συμμετάσχει σε ένα ομοσπονδιακό δίκτυο τόσο για την κατασκευή ενός ισχυρού, γενικευμένου μοντέλου, όσο και για να το εκθέσει στα τοπικά του δεδομένα και να βελτιώσει την επίδοσή του πάνω σε αυτά.

0.4 Επίλογος

Σε αυτήν τη διπλωματική εργασία εξετάσαμε τη συνεργατική εκπαίδευση αναδρομικών μοντέλων μηχανικής μάθησης από ένα δίκτυο νοσοκομείων για το πρόβλημα της έγκαιρης πρόβλεψης του κινδύνου θνησιμότητας στη ΜΕΘ, χρησιμοποιώντας ζωτικές ενδείξεις και εργαστηριακές εξετάσεις με τη μορφή χρονολογικών σειριακών δεδομένων. Τα δεδομένα μας προήλθαν από τη Συνεργατική Ερευνητική Βάση Δεδομένων eICU, η οποία απαρτίζεται από πραγματικά δεδομένα μονάδων εντατικής θεραπείας των ΗΠΑ, επιτρέποντάς μας να εξετάσουμε ένα ρεαλιστικό ομοσπονδιακό δίκτυο νοσοκομείων με στατιστική ανομοιογένεια. Σχεδιάσαμε μια σειρά από πειραματικά σενάρια με στόχο να παρατηρήσουμε την επίδοση διαφορετικών μοντέλων και αλγορίθμων ομοσπονδιακής μάθησης υπό διαφορετικές συνθήκες. Τα αποτελέσματά μας έδειξαν ότι το μοντέλο GRU ενσωματώνεται πιο αποδοτικά στο πλαίσιο της ομοσπονδιακής μάθησης για το συγκεκριμένο πρόβλημα, ενώ τα μοντέλα ομοσπονδιακής μάθησης πετυχαίνουν σημαντικά καλύτερες επιδόσεις από τα τοπικά μοντέλα και πλησιάζουν αρκετά εκείνες της ιδανικής, αλλά συνήθως ανέφικτης λόγω της ιδιωτικότητας των δεδομένων, κεντρικής μηχανικής μάθησης. Έπειτα, παρατηρήσαμε ότι ο αλγόριθμος FedProx, που στοχεύει στη βελτιστοποίηση της ομοσπονδιακής μάθησης σε τοπικό επίπεδο, έχει μεγαλύτερη αποτελεσματικότητα σε δίκτυα με την ύπαρξη νοσοκομείων μεγάλης επιρροής και ακραίας υποεκπροσώπησης της θετικής κλάσης, ενώ οι αλγόριθμοι βελτιστοποίησης στο επίπεδο του κεντρικού εξυπηρετητή συνεισφέρουν περισσότερο σε δίκτυα με την ύπαρξη νοσοκομείων μικρής επιρροής και σημαντικών πληροφοριών για την θετική κλάση που υποεκπροσωπείται. Τέλος, είδαμε πως ένα νοσοκομείο έχει σημαντικό κίνητρο να συμμετάσχει σε ένα ομοσπονδιακό δίκτυο, καθώς το μοντέλο που εκπαιδεύει παρουσιάζει καλύτερη επίδοση πάνω στα τοπικά του δεδομένα σε σχέση με άλλες εναλλακτικές, που διασφαλίζουν σε ένα βαθμό την ασφάλεια και ιδιωτικότητα των δεδομένων.

Chapter 1

Introduction

In the digital era, an exponential upsurge in data generation has transformed the landscape across various domains. Sensors, trackers and connected devices capture not only immense volumes of information, but also the temporal aspect of the data, making it inherently time series in essence. This proliferation of time series data presents unprecedented opportunities for extracting valuable insights, identifying patterns, and making data-driven decisions that can drive innovation and advancements across numerous industries. The distributed nature of this data, often residing in remote devices and endpoints, calls for innovative approaches to harness its vast potential.

In response to this, traditional centralized approaches face significant challenges. A central server may be unable to meet the computational and memory requirements for processing massive quantities of data, leading to performance degradation, slowdowns, or even failures. Then, the amount of communication needed introduces additional matters that should be considered. The transmission of raw data from remote devices to a centralized location raises concerns regarding data privacy and security, as sensitive information may be exposed. Moreover, network issues during data transfer may cause increased latency, delays and interruptions in the overall workflow. As a result, federated learning has emerged as a promising solution, leveraging the decentralized nature of the data and enabling collaborative model training while preserving privacy, reducing communication overhead and parallelizing the computational workload across multiple local devices.

One domain that could be greatly benefited from the federated learning approach is healthcare, which generates data at an unprecedented rate. The availability of electronic health records (EHRs), containing a wealth of patient information in the form of multivariate time series (MTS), and the accumulation of healthcare data from multiple sources present tremendous opportunities for improving healthcare delivery, enhancing clinical decision-making, and advancing medical research. By aggregating data from various medical centers, researchers and healthcare professionals can gain deeper insights into diseases, treatment effectiveness, and patient outcomes. However, the integration and analysis process is complicated by data heterogeneity, due to variations in data formats, collection methods and patient populations. Moreover, privacy concerns, data security and regulatory compliance pose significant hurdles when for data aggregation.

Federated learning offers a compelling framework to address the challenges of health-

care data collaboration. By decentralizing the model training process, federated learning allows healthcare institutions to retain control over their local data and collaborate on developing state-of-the-art ML models, without sharing sensitive patient information. This approach has the potential to effectively capture the nuances and characteristics of heterogeneous local datasets, while it significantly reduces data privacy and security risks. The ML models constructed in the context of federated learning often display improved accuracy and generalizability, as they have been exposed to a diverse dataset, comprising data from various sources. Overall, the application of federated learning in healthcare holds great promise for revolutionizing the field and advancing patient care.

In this thesis, we investigate the application of federated learning in the healthcare sector, with a focus on addressing the early ICU mortality risk prediction task with the utilization of MTS data from multiple hospitals. In Chapter 2, we formulate this classification problem and describe the objectives of this study. Then, we introduce some fundamental theoretical concepts that are prevalent throughout this thesis. In Chapter 3, we discuss about the characteristics of time series and how to handle them efficiently. In Chapter 4, we present an overview of deep neural network architectures, focusing on their use with time series data. In Chapter 5, we introduce the federated learning framework, examining its basic workflow, different federated learning algorithms and compatible privacy-preserving and security mechanisms. Afterwards, in Chapter 6, we present some related works to this study, before we turn our focus on our experiments. In Chapter 7, we present the eICU Collaborative Research Database and analyze our cohort selection criteria and data preparation methodology. In Chapter 8, we elaborate on the federated learning scenarios for our experiments, and lay out our evaluation strategy. In Chapter 9, the results of our experiments are presented and discussed in detail. Finally, in Chapter 10, we recapitulate the main conclusions of this study and propose some future research directions.

Chapter 2

Problem Formulation

The problem studied in this thesis is the *early ICU mortality risk prediction*, leveraging data from multiple healthcare centers. Patients admitted in intensive care units typically exhibit a critical health condition that requires continuous medical care and monitoring. Early identification of patients at high risk of mortality within a specific time frame after ICU admission plays a crucial role in improving patient results. By developing accurate, forward-looking predictive models, healthcare providers are enabled to take action promptly to prevent adverse outcomes. Therefore, early ICU mortality risk prediction is a significant medical task with vast and impactful potential.

We formulate the task of predicting ICU mortality as a binary classification problem, with multivariate time series, where the patients who died during their ICU stay constitute the positive group (output = 1) and the patients who got discharged constitute the negative group (output = 0). The time point indicating the mortality event or discharge is defined by the *ICU discharge offset* variable, while the beginning of the ICU stay is defined as the first vital signs measurements after the ICU admission record. Since this study focuses on *early* prediction, we consider an observation window during the first hours of a patient's ICU stay. It consists of 24 hours of vital signs (7 variables) and laboratory tests (16 variables) after the beginning of the ICU stay. The MTS data from this observation window are extracted and utilized for training and evaluation of the predictive models. The resulting algorithm indicates the mortality likelihood of a patient during the 48-hour period after the prediction time, which is at the 24-hour mark of the ICU stay.

To address the early ICU mortality risk prediction problem, deep recurrent neural network architectures are employed across different ML setups. The typical centralized machine learning (CML) approach assumes that patient data from the participating healthcare centers are collected and processed on a central server, enabling the development of more generalizable models. However, certain challenges are presented with concerns over data privacy, security and scalability. Its direct alternative is the local machine learning (LML) approach, which assumes no data sharing among the participants, training a ML model on each healthcare institution on its local dataset. Even though this method reduces privacy and security risks by eliminating the communication of sensitive information, the obtained models are often subject to bias and limitations of the local datasets. Counteracting the shortcomings of the aforementioned approaches, federated learning (FL) is a promising paradigm that allows the collaboration of multiple

medical centers on training a global predictive model, while keeping the sensitive patient data localized. Instead only local model parameters are shared and aggregated at a central server, to construct the global model. Thus, this decentralized approach utilizes the statistical power of multiple local datasets to develop generalizable models in a privacy-preserving manner.

The goal of this study is to evaluate the performance of state-of-the-art FL algorithms on the early ICU mortality risk prediction task, using MTS data, with a realistic, heterogeneous setting of participating hospitals. Initially, we aim to explore how well different RNN architectures are integrated into the federated learning setting, with respect to this problem, while also evaluating the ability of the FL models to generalize on a foreign test set, compared to the CML and LML approaches. Then, we recreate FL environments with the participation of hospitals with 'extreme' data distributions and analyze the sensitivity of each FL algorithm under challenging conditions for collaborative training. Finally, we investigate the impact of participation in federated learning, by focusing on specific hospitals and how their engagement in a FL training environment may improve their predictive performance on their local data. Overall, we conduct a thorough study of the application of federated learning methods on early prediction of ICU mortality risk with a real world, multi-center database, showcasing the potential of federated learning in the healthcare domain.

Chapter 3

Time Series

This chapter provides an overview of time series. First, the concept of time series and their main characteristics are presented. Afterwards, we examine standard preprocessing techniques for handling common time series problems. Then, we discuss about the role of time series data within the healthcare domain.

3.1 Introduction to Time Series

With the advent of Internet of Things (IoT) devices, sensors, and social media platforms, vast amounts of time series data are being generated every second. A *time series* is a sequence of data instances indexed in time order, where each data instance represents an observation or measurement that corresponds to a specific moment or period of time. These measurements are referred to as metrics, when they are collected at regular time intervals, or as events, when the observations are unevenly spaced over time [39]. Time series data has numerous applications in a wide range of fields, from finance and healthcare to engineering and environmental science.

The unprecedented amount of data being generated in today's world has led to an increasing need for an advanced set of methods for extracting useful statistics and characteristics of time series data, referred to as *time series analysis*. Depending on the nature of the application domain, the primary goals of the analysis vary. In the context of signal processing, the analysis aims first and foremost at signal detection, while within data mining and pattern recognition, the primary objective could be clustering, or anomaly detection. However, most applications focus on constructing a model to predict the future values of a time series by studying the relationship of previous values with each other, a procedure which is called *time series forecasting* [18].

3.2 Characteristics of Time Series

There are two main categories of time series data, depending on the amount of observed variables. *Univariate time series* are observations of a single variable or feature over a time period. Because they have only one dimension other than time, they can be visualized easily, leading consequently to easier overall analysis. On the other hand, *multivariate time series (MTS)* are observations of two or more variables over time [40].

The multiple features make the analysis of the time series more challenging, however a successful analysis of the interdependencies among the the variables can give better insights about the nature of the data. In healthcare, both types of time series can be utilized. A healthcare professional could focus on a single aspect of a patient's health condition, such as their heart rate, or they could analyze their condition from various angles, assessing the underlying patterns of a combination of patient data (Figure 3.1).

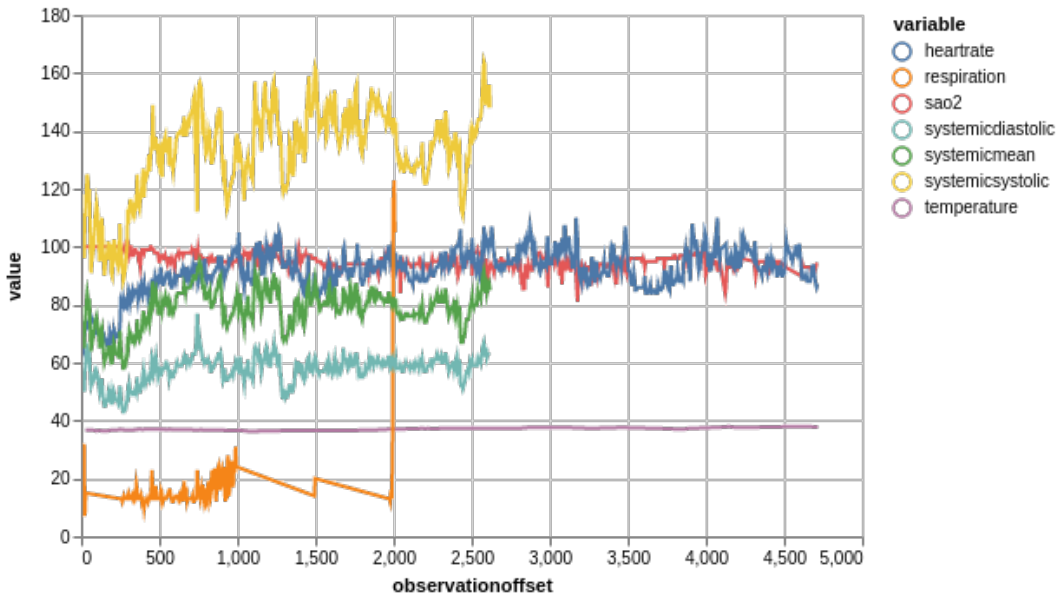


Figure 3.1. *Patient vital signs during the first hours of an ICU stay [1]*

However, in both cases, in-depth understanding of the characteristics of a time series data is crucial for developing effective models for classification, prediction and other tasks. Below, we present certain basic characteristics of a time series:

- ***Trend*** is a pattern that indicates the tendency of a time series to increase or decrease over a long period of time [40]. A pattern of movement towards higher values for a time series is called an *uptrend*, while a downwards trend is called a *downtrend*. When the time series values increase or decrease with a constant rate, the trend is *linear*, while a *nonlinear* trend is characterized by a changing rate. Early detection of trend is important for many applications in various domains. For instance, in clinical decision-making, if the heart rate or respiratory rate of a patient display a trend, then this could indicate the deterioration of their condition to their healthcare provider.

- ***Seasonality*** is a common time series characteristic that refers to the repeating pattern of periodic fluctuations, occurring at regular intervals over time [40]. Depending on the problem at hand, a variety of factors can cause seasonality. For instance, in healthcare, a patient could be administered a specific drug at a certain time each day, causing lower-than-normal measurements of his respiratory rate within a short period of the drug administration. Consequently, seasonality, when detected, is an important factor that should be taken into account for forecasting.

- ***Cycle*** is a time series feature that refers to a repeating pattern that occurs over a period of time, similar to seasonality. However, there are some key elements distinguishing cycle from seasonality. Cycles describe medium-term changes in the time series, in contrast to the short-term seasonal patterns [40]. Then, if the period of a pattern is stable and connected to a specific aspect of the time frame, this behavior is characterized as seasonal, whereas an irregular period indicates cyclical, variable behavior. The detection and modeling of cycles in time series data has various applications across different domains and, thus, it is a crucial aspect of time series analysis.

- ***Stationarity*** is a key property of a time series that refers to the constancy of its statistical properties over time, thus indicating neither trend nor seasonality. In other words, the mean and variance of a stationary time series do not fluctuate. There are two types of stationarity: *strict stationarity*, which requires that the joint distribution of any subset of the time series remains the same regardless of the observation time points, and *weak stationarity*, which requires that the mean value of the time series is constant and the auto-covariance of two observations depends only on the chronic difference of the observations and not the observation time intervals themselves [41]. A stationary time series is ideal for stable analysis and modelling, because the underlying patterns and relationships are not obscured by trends or seasonality.

- ***Outliers*** are irregular data points in a time series that deviate from the expected from the expected pattern of the data over time [40]. These random fluctuations do not indicate a specific pattern by repeating themselves over time and, as a result, they cannot be predicted by any statistical technique. The cause of their sudden appearance may be an external unpredictable factor, such as the rapid increase of the patient admissions to ICUs after the outbreak of a contagious disease or an error during data collection. Outliers pose as a major challenge for time series analysis, since, without careful treatment, they may lead to false results.

The aforementioned time series characteristics are equally important in studies of both univariate and multivariate time series. Nevertheless, the nature of MTS data showcases the need for considering even more aspects of a time series during the analysis process, in order to gain a multifaceted understanding of the subject and construct more robust models [42]. Thus, we point out some key aspects that are more often present in MTS data:

- ***Interdependencies*** among the observed variables require a good understanding, in order to design accurate and efficient models. Studying the cross-correlation between each pair of variables may give fruitful insights about the underlying relationships and indicate the need for a multivariate transformation of the data, such as using principal component analysis (PCA) [42] [43]. For example, the cause-effect relationship of the administration of a drug to a patient and the subsequent changes

to their vital signs should be taken into account when assessing a patient's medical condition.

- ***High dimensionality*** is, as expected, more common in multivariate time series data than univariate. A higher dimensional space leads to increased complexity and makes the analysis and visualization of the data more challenging [44]. It is important to use appropriate tools for dimensionality reduction and visualization or select a subset of features accurately, to effectively reduce the overall complexity and retain as much useful information as possible.
- ***Missing values*** is a major obstacle in time series analysis. Because MTS data comprise of observation of multiple variables, the problem of missing values occurs more frequently. By using the appropriate technique to handle missing values, depending on the task, the results of a time series analysis could be considerably better and more insightful [45]. Regarding a patient's hospital stay, different methods to handle missing vital sign measurements may lead to quite divergent conclusions about his health status.

3.3 Time Series Preprocessing Techniques

In the previous section, we presented some key time series features that should be taken into consideration before the phase of time series analysis and forecasting commences. These characteristics require specialized preprocessing techniques to be effectively utilized. Thus, the preprocessing phase should be carefully designed and implemented for the subsequent success of the analysis and modeling stages. Below, we present some basic preprocessing techniques that deal with each of the aforementioned time series characteristics.

3.3.1 Time Series Decomposition

Decomposition of a time series is a common preprocessing method that aims to the isolation and analysis of individual time series components. Usually, a time series is decomposed to three underlying components [4]. The first is a combination of trend and cycle into a *trend-cycle component* (usually referred as trend), which captures the overall trend and medium and long-term patterns of the time series. Then, the *seasonal component* consists of the short-term regular changes to the time series. Finally, the *remainder component* contains anything that cannot be explained by the other two components [4].

The classical time series decomposition method was introduced in the 1920s and forms the basis for many modern decomposition techniques [4]. The first step of this method is the application of a *moving average* filter, to produce an estimation of the trend-cycle component. The equation of a moving average filter of order m can be written as

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j},$$

where $\hat{T}_t$ is the estimate of the trend-cycle component at time t by calculation of the average values of the time series y_t within k periods of t and $m = 2k + 1$ [4]. This technique eliminates any seasonal patterns and some of the randomness introduced by noise to produce a smooth trend-cycle component [46].

There are two models of classical decomposition: the *additive model*, when the level of the time series does not have an effect on the variations around the trend, and the *multiplicative model*, when the trend and the level of the time series are proportional. For the additive model

$$y_t = \hat{T}_t + \hat{S}_t + \hat{R}_t,$$

where $\hat{T}_t$, $\hat{S}_t$ and $\hat{R}_t$ are the trend-cycle, seasonal and remainder component respectively, the seasonal component is calculated by averaging the values of the de-trended series, $y_t - \hat{T}_t$,for each defined season, and the remainder component is the residue of the other two components, $y_t - \hat{T}_t - \hat{S}_t$. Similarly, for the multiplicative model

$$y_t = \hat{T}_t \times \hat{S}_t \times \hat{R}_t,$$

the component calculations are the same, but with divisions instead of subtractions to isolate the time series components. Figure 3.2 shows an example of the use of additive decomposition method on a time series of the total retail employment across the US.

However, the classical decomposition methods have significant drawbacks, such as the over-smoothing of the trend-cycle component and the inability to estimate it for the first and last observations, the assumption that seasonal patterns remain unchanged over time and the mishandling of irregular time series observations over small periods [4]. As a result, modern methods have been developed to tackle with these problems. Some of the most popular techniques that are widely used by statistics agencies are the *X-11 Method* [47], originating from the US Census Bureau and further developed by Statistics Canada, SEATS (Seasonal Extraction in ARIMA Time Series) Method [47], developed by Bank of Spain, and the STL (Seasonal and Trend decomposition using Loess) Decomposition, developed by R.B.Cleveland et al. [48], and their variations.

3.3.2 Handling Outliers

As explained earlier, outliers are observations that deviate from the overall behavior of the series, which is dictated by the majority of the observations. As such, it is vital that they are detected with precision and handled carefully, in order to limit their detrimental effect on the analysis of the time series. One approach on the detection of outliers is using statistical methods, namely the *Z-Score*, the *Modified Z-Scores*, the *Median Absolute Deviation (MADe) method* and the *Tukey boxplot method* [49]. These techniques use the distribution of the data and their statistical properties to determine a range of values that would be accepted as normal, pointing out every observation that falls outside of it as an outlier.

Once the outliers are detected, the proper course of action should be considered re-

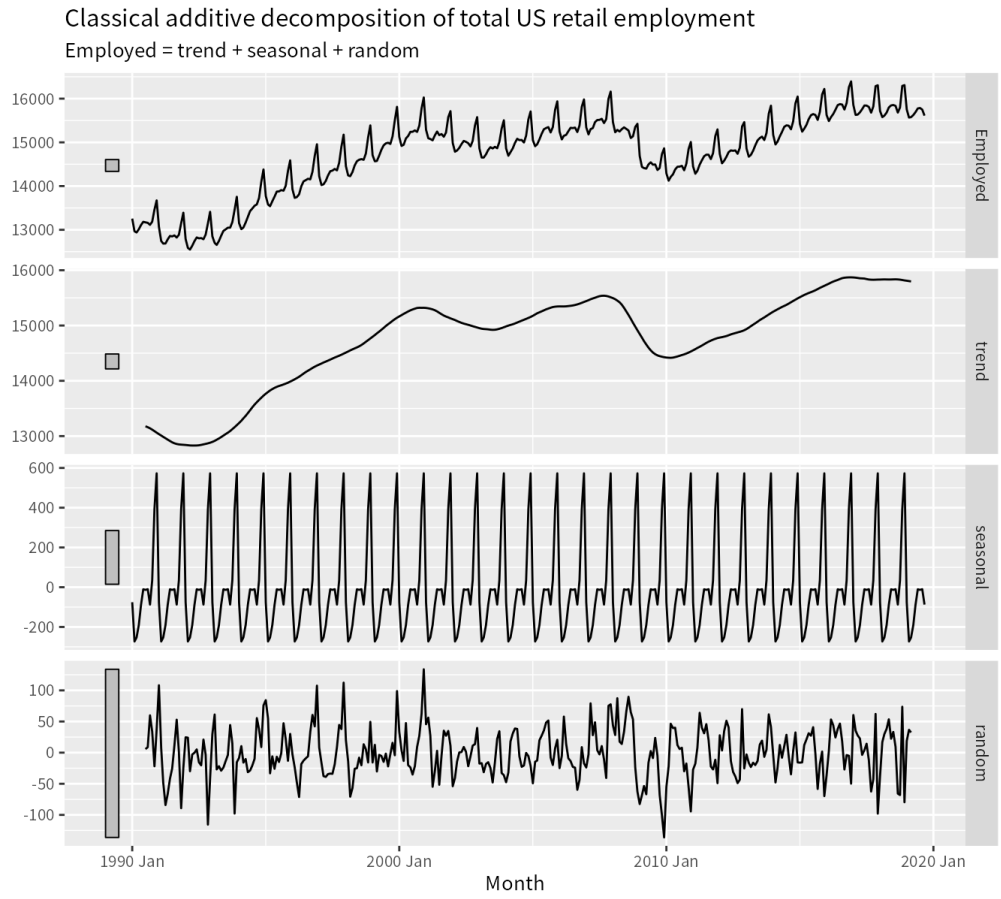


Figure 3.2. *US retail employment series and the additive decomposition components [4]*

garding how to handle them. One approach is to exclude outliers from the dataset. There are two common methods to exclude outliers: *trimming*, where the outliers are simply discarded, and *Winsorising*, where the outliers are replaced with the nearest observation that is not considered an outlier [19]. However, if the number of available observations is limited, these techniques may lead to significant loss of information.

Another approach to deal with outliers for time series analysis is to use analysis methods that are resistant to the effects of outliers. Using robust time series analysis techniques minimizes the impact of outliers on the results, while they are not excluded with the techniques that were previously mentioned. For instance, regression types, such as standard square error loss, have desired properties if their underlying assumptions are true, which outliers may violate, thus obscuring the results. Robust regression [20] reduces the outliers' contributions to the results by using robust alternatives, such as the Huber loss function [50].

3.3.3 Feature Selection, Engineering & Scaling

Multivariate time series data usually involve multiple variables with different units, scales and ranges. Therefore, the high dimensional space and scale differences of MTS data pose as major challenges, in terms of computational cost and performance, for time series analysis tasks and model training. As a result, it is important to address them in

the preprocessing phase [51].

Feature selection is a method for selecting a subset of the available time series features, thus improving the predictive performance of ML models by avoiding overfitting and going into a lower dimensional space, often with lower computational costs and fewer trainable ML model weights [21]. Moreover, the results could be an easily interpreted representation of the objective function [52], thus providing a better understanding of the nature of the data and their importance for the task at hand [21]. However, feature selection is a costly procedure that should be carefully implemented, in order to limit the loss of useful information and approximate the assumed optimal performance, which utilizes all the available information [51].

Another solution for the high dimensionality problem and for better usage of the time series features within the ML framework is *feature engineering*. This method involves studying the original features of a time series and, by using domain knowledge, constructing new features from the pre-existing ones [22]. These new features represent different, more complex aspects of the time series. Since the representation of the feature vector has a direct effect on the performance of ML models [22], these constructed features could significantly improve the quality of the results. Some typical methodologies to engineer useful features are clustering algorithms [53], PCA [43], category encoders [54] (such as one-hot encoder), group aggregated values and mathematical transformations [55].

A common feature engineering technique is *feature scaling* (or *data normalization*), which is an essential step in preprocessing multivariate time series data. Variables with different scales may deteriorate the performance of a machine learning algorithms, since some may unjustly dominate over others causing bias in the model. Through feature scaling, the MTS data are transformed such that each variable lies within a specific range or scale, thus eliminating any phenomena of bias. Moreover, deep neural network training is expedited, because gradient descent convergence is achieved considerably faster after data normalization [56]. Two of the most common feature scaling methods are *min-max scaling*, which re-scales each variable within a range $[a, b]$, and *standardization* (or *z-score normalization*), which transforms each variable such that it has zero-mean and unit-variance.

3.3.4 Handling Missing Values

Missing values are a common phenomenon in time series data, often due to sensor malfunctions, data corruption or errors during data collection [57], leading to biased results and inaccurate predictions. Consequently, it is crucial to deal with missing values effectively during the preprocessing stage. There are several sequential *imputation* methods for missing observations. One approach is to delete data that contain missing values, however this may lead to significant information loss [4]. Other approaches focus on replacing the missing observations by employing information from the existing data. Some common techniques are replacing a feature's missing values with the mean value, with the value of the median observation, with the closest previous observed value or with the

closest following observed value [4].

An alternative approach is *interpolation*, which estimates missing values by using the observations around them. Some common interpolation techniques utilize adjacent data observations to make this estimation, by using a linear function (linear interpolation), low-degree polynomial functions (spline interpolation) or higher-degree polynomial functions (polynomial interpolation). The simplicity of the interpolation techniques favour them over imputation methods, when the missing values are sporadic.

3.4 Time Series in Healthcare

With the widespread adoption of electronic health records (EHRs) and registries, vast amounts of patient time series data are generated within the healthcare sector. They inform on genetic and lifestyle health risks, signal the onset or presence of diseases, indicate the time and stage of diagnosis and record the drafting of treatment plans and their outcomes [5]. This has motivated further research on how time series data can be utilized to provide solutions to many healthcare-related problems [58].

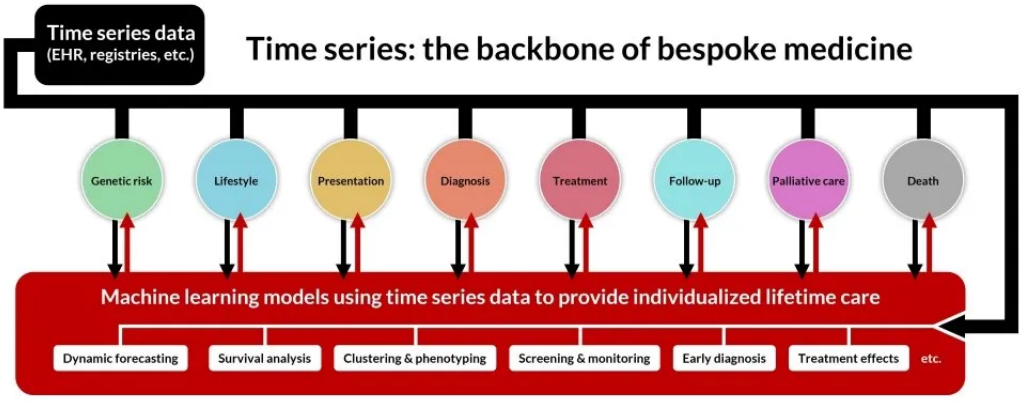


Figure 3.3. *Development of machine learning models within the healthcare domain, using time series [5]*

By employing time series analysis methods to recognize trends, patterns, and anomalies in patient data, and by developing powerful machine learning models, healthcare providers can make more accurate diagnoses, predict the trajectory of patients' medical conditions and their outcomes, as well as shape personalized treatment plans, tailored to each individual patient's needs [59]. Gradually, machine learning could be the cornerstone for developing long-term comprehensive patient management programs that monitor the patient's life and adjust accordingly [5]. Potential benefits of such practices can be also pointed out for the overall operation of a medical center. Time series analysis techniques and ML models can help medical centers optimize resources allocation, including staff, equipment and supplies, to meet patient needs and minimize the spread and impact of infectious diseases with early detection [60].

However, time series introduce some unique challenges to model development. Some originate from the nature of the data, such as missing values and outliers which we

discussed earlier, while others are related to the sensitive nature of the healthcare domain. Healthcare problems pose some additional requirements for the deployment of ML models to tackle them. The outputs of the models should be actionable, informative and reliable, for a healthcare professional to take them into account and act on them [5].The models should also offer uncertainty estimates, which may lead a healthcare professional to give an analogous weight to their suggestions when assessing a situation. Moreover, the models' suggestions should ideally be interpretable, in order to assist clinicians gain new insights and explain to their patients the reasons behind a specific course of action [5].

Consequently, time series data in healthcare constitute an exciting research domain. Combined with the computational capabilities of machine learning models, healthcare professionals gain a deeper understanding of individual health and disease trajectories, thus being able to aid patients in their health-related decision making and designing effectively their treatment plans [5].

Chapter 4

Deep Neural Networks

This chapter contains an overview of *deep neural networks* (DNNs). First, it presents some fundamental concepts to better understand DNNs, such as machine learning and artificial neural networks. Then, it delves deeper into feedforward and recurrent neural network architectures, discussing their characteristics and process of learning.

4.1 Central Concepts

4.1.1 Machine Learning

Machine learning (ML) is a subfield of Artificial Intelligence (AI) that focuses on understanding and developing models that enable computer systems to "learn", which is to improve their performance on a set of tasks by effectively utilizing data, according to Mitchell et al. [23]. A model is constructed using a machine learning algorithm, which is trained on a set of data. During the training process, the algorithm learns underlying patterns and relationships in the training data and, then, uses this gained experience to make a prediction on new data

There are three main categories of approaches to machine learning, depending on the provided feedback from the training data. *Supervised learning* is a machine learning paradigm, where the available data consist of both a set of inputs for the model (features) and the desired outputs (labels) [61]. In this scenario, supervised learning algorithms attempt to optimize an objective function, which, subsequently, can be used to predict the label associated with a set of features of a data point [62]. When the data do not contain desired outputs for the learning process, then the respective learning algorithms fall under the *unsupervised learning* paradigm. The goal of unsupervised learning is to explore the nature of the data itself and discover underlying patterns that can be used to group similar data instances into clusters or project the data into a lower-dimensional space, while retaining as much useful information as possible. Finally, the *reinforcement learning* paradigm is used in dynamic environments, which an intelligent agent navigates according to a set of rules, with a specific goal. The agent aims to maximize the rewards they earn with their actions, which is a form of feedback in this paradigm [63]. This thesis is concerned with a supervised learning classification problem.

4.1.2 Artificial Neural Networks

Artificial neural networks (ANNs), commonly referred as *neural networks* (NNs), are computing systems designed after the structure and function of biological neural networks. Neural networks consist of layers of interconnected computing cells, referred as neurons or processing units, while the weighted connections among them are called links or synapses [24]. During training, each neuron takes its weighted sum of the inputs from the previous layer, processes it using an activation function and, then, passes the output to the neurons of the next layer it is connected to (Figure 4.1). To optimize the performance of the neural network, in a supervised learning scenario, the weights of the connections are adjusted to minimize the difference between the predicted outputs and actual outputs of the training data.

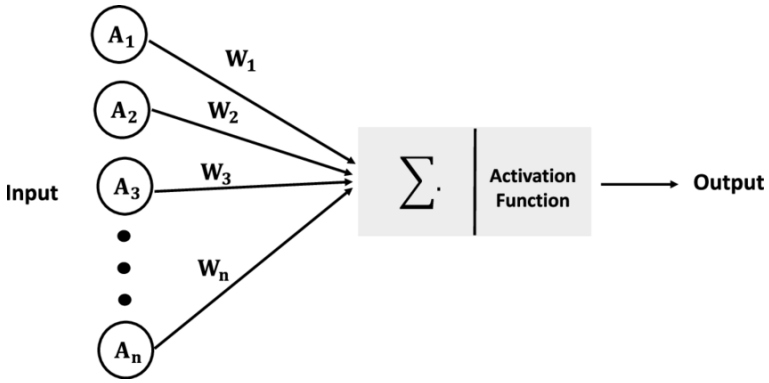


Figure 4.1. *The structure of an artificial neuron [6]*

The basic concept of ANNs has inspired *deep learning (DL)*, a class of machine learning algorithms that employ multiple layers in the network to extract higher-level features from the input data [25]. Each layer of these deep neural networks learns to transform its input into more abstract and complex representations, thus identifying subtle patterns and relationships that would otherwise be extremely difficult or impossible to detect. Depending on the task, machine learning models are based on one of two types of networks, feedforward neural networks and recurrent neural networks, which will be presented in the following sections of this chapter.

4.1.3 Learning via Training

As stated previously, a neural network adjusts the synapses weights based on the deviation of the predictions from the desired outputs of the training data, in a supervised learning setting. Nevertheless, it is important to dive deeper into the training procedure and understand exactly how the model evolves to optimize its performance.

The first phase of the training process is called the *forward pass*. The features of a data instance are passed as input to the first layer of the network. Following the workflow mentioned earlier, each neuron processes its input and passes its new output as input to the corresponding neurons of the next layer. When the last layer of the network receives its input, it produces a prediction for this data instance [2]. At this time, a selected

loss function is employed to quantify the error of the prediction in regard to the actual output, an evaluation metric for the performance of the network. Then, the second phase of the training, called *backpropagation*, commences. Backpropagation is a commonly used algorithm for computing the gradient of a prediction-label pair, with respect to the loss function. However, this term is often used to describe how the gradient is utilized afterwards, such as by stochastic gradient descent [64]. In reverse order, according to the chain rule, the gradients of all the layers up to the input layer are computed [2]. The gradients of each layer are used to update the weights of the layer's nodes, in order to decrease the total loss.

For each sample of the training set, the forward pass and backpropagation are used interchangeably, updating the network weights with the gradients calculated by backpropagation. This procedure is repeated for a predefined number of *training epochs* or until convergence is achieved. Although the final model could have excellent performance on the training data, the goal is produce a model capable to *generalize* beyond the training set [2]. As a result, it is important to overcome the challenge called *overfitting*, that is a model performing excellent on the training data, but poorly on an unknown dataset. A method to overcome this problem is to calibrate the ML model properly with *regularization techniques*, either on the loss function, such as L1-Regularization and Dropout, or on the time dimension, by using early stopping criteria on the training [2].

4.2 Feedforward Neural Networks

The *feedforward neural network* (FNN) is the first and simplest type of artificial neural network [65], where the information flows only forward, from input to output, as there are no recurrent connections [66]. Successive layers are fully connected, meaning each neuron is connected to all the neurons of the following layer. Depending on the number of layers that constitute the network, FNNs are characterized as single-layer or multi-layer FNNs. A *single-layer FNN* (also called single-layer perceptron) consists of a single layer of neurons, where the input is multiplied by a single set of weights and passed directly through an activation function to produce the output. Marvin Minsky and Seymour Papert proved in their monograph titled *Perceptrons* that it was impossible for a single-layer perceptron to learn the XOR function [67], which showcased the inability of single-layer FNNs to learn non-linear relationships. On the other hand, a *multi-layer FNN* (also called multi-layer perceptron) contains one or more hidden layers between the inputs and the output layer, addressing the non-linearity limitations of single-layer models [2]. The transformations through the hidden layers enable the model to learn complex patterns in the input data.

Deep learning refers to the number of hidden layers in a neural network. Each hidden layer transforms the input into more abstract representations, thus allowing the network to gain a "deeper" understanding of the data. Except from the size of each hidden layer and the "depth" of the network (the amount of hidden layers and output layers for FNNs), a DNN organizes the transformations and the derived representations of each layer on its own [25] during training. Even though the resulting model learns composite underlying

patterns of the input data, often allowing it to perform exceptionally in the objective task, its mathematical manipulations of the input and the respective abstract representations are hardly impossible to be interpreted by a human.

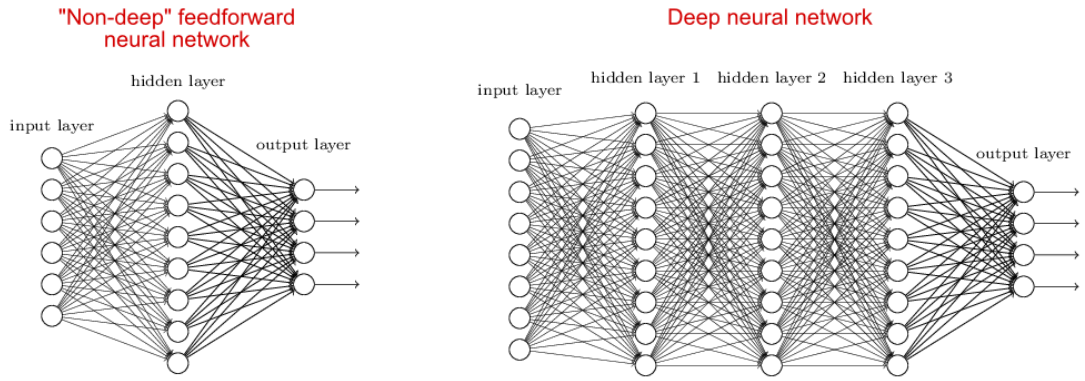


Figure 4.2. *A shallow multi-layer FNN with 1 hidden layer and a deep multi-layer FNN with 3 hidden layers [7]*

4.2.1 Convolutional Neural Networks (CNN)

There are two major challenges related to the fully-connected layers of deep feedforward neural networks. First, an increase in the size of the input data subsequently leads to an increase in the number of trainable weights and the overall complexity of the deep FNN. Then, the "full-connectivity" makes the deep FNN prove to overfitting on the training data. A regularized version of deep FNNs that deals with these problems is the *convolutional neural network* (CNN), a class of deep neural network that is widely used in image recognition and processing [68], with applications for time series tasks as well [69].

The most fundamental component of the CNN architecture is the *convolutional layer*, which transforms its input to an abstract representation called *feature map* [64]. Similarly to the cortical neurons that inspired them, the neurons of a convolutional layer receive input from a restricted area of the previous layer that constitutes their *receptive field*. Then, the dot product of this input with a convolutional kernel is calculated to generate a representation of the input. For two-dimensional tensors f and g, the mathematical expression of convolution [2] is:

$$(f * g)(i,j) = \sum_a \sum_b f(a,b)g(i-a, j-b).$$

By repeating this procedure for all the layer's neurons, the feature map is constructed and, subsequently, used as input to next layer of the network. The convolution approach requires significantly less parameters to process the data than deep FNNs, thus allowing the construction of "deeper" neural networks [70], capable to learn increasingly complex patterns and representations of the input. The convolutional layers are usually intertwined with *pooling layers*, with the goal to reduce the dimensions of the data by aggregating the outputs of a cluster of neurons into a single cluster [2]. A pooling layer

is characterized as *global*, when it is applied on all the neurons of the previous layer's feature map, and *local*, when it combines small clusters of neurons. Then, the two most popular pooling aggregation methods are *max pooling* and *average pooling*, where the representative value of a cluster of neurons is the max or the average value respectively. Finally, the last layers of a CNN are often fully-connected layers, referred as *dense layers*.

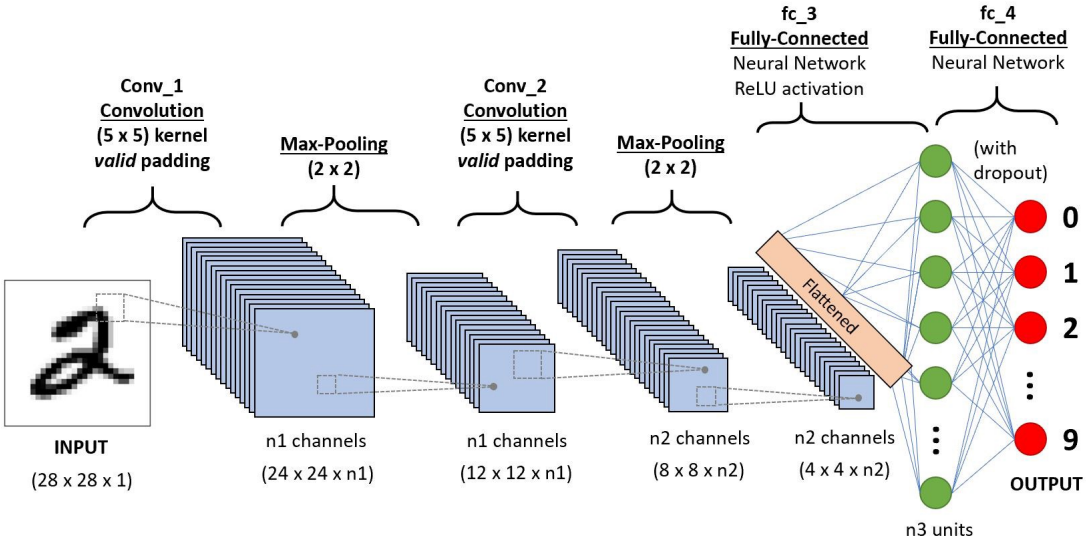


Figure 4.3. *A CNN for handwritten digit image classification [8]*

4.2.2 Temporal Convolutional Networks (TCN)

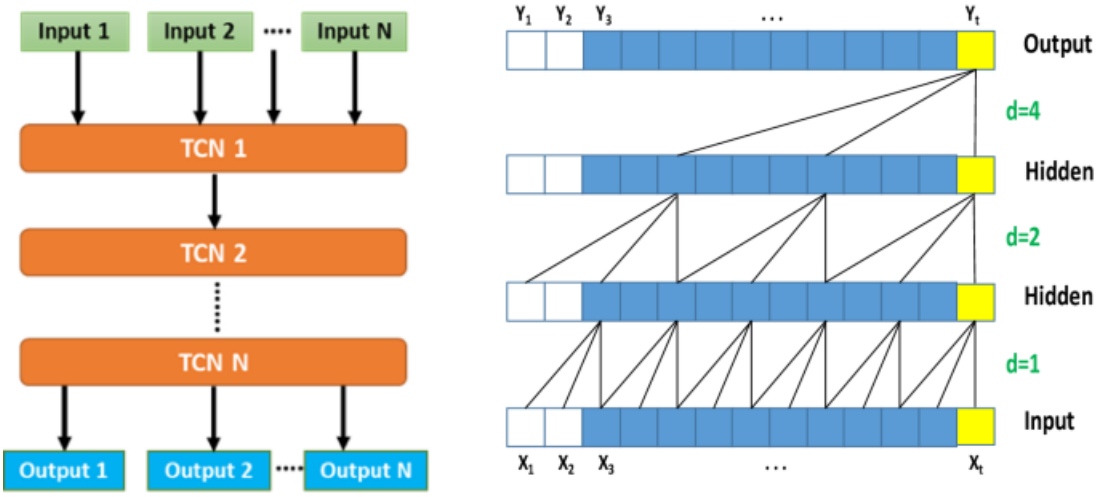
Following the paradigm of the convolutional neural networks, *temporal convolutional networks* (TCNs) were recently proposed to explore long-range temporal patterns, using a hierarchy of temporal convolutions [71]. Similarly to a recurrent neural network (presented later), a TCN can map an input sequence of any length to an output sequence of the same length, by using a one-dimensional, fully convolutional network architecture [9]. One other key characteristic of TCNs is the use of *causal convolutions*, which prevents information "leakage" from future observations to the past [72]. By applying zero-padding at the left side of the input tensor, an output at time t is convoluted only with elements of the previous layer from time t or earlier, thus ensuring causal convolution.

One desirable feature of temporal convolutional networks is the ability to effectively deal with long time series. However, a simple causal convolution can only look back at a history of linear size, depth-wise in the network [72]. To tackle this challenge, the *dilated convolution* is employed, which enables an exponentially large receptive field [73]. The dilated convolution operation on a one-dimensional input sequence s of a parameter p is defined [9] as:

$$(p *_d f)(s) = \sum_{i=0}^{k-1} f(i) p_{s-d.i},$$

where d is the dilation factor, k is the size of the filter f and $s - d.i$ accounts for the

direction of the past. The dilation factor corresponds to a fixed step between adjacent filter taps. Larger dilation factors and larger filter sizes allow the expansion of the receptive field, as illustrated in Figure 4.4. By stacking multiple TCN layers on top of one another, the construction of deep learning models is enabled, while the characteristics of TCNs make them a strong candidate for complex time series problems [74]. The convolution architecture allows the parallel process of long input sequences, leading to lower computational time and better memory usage than RNNs, which process the input data in a sequential manner [74].



(a) Proposed DNN with stacked TCN layers

(b) A typical TCN layer

Figure 4.4. *Architecture of a TCN with causal convolution and different dilation factors [9]*

4.3 Recurrent Neural Networks

A *recurrent neural network* (RNN) is a class of artificial neural networks that captures the dynamics of input sequences via recurrent connections in the network [2], which create a cycle that allows the output of some nodes to influence the subsequent input of the same nodes. The basic RNN architecture consists of the input layer, a hidden layer and the output layer. As shown in Figure 4.5, recurrent neural networks are unrolled over time steps, corresponding to the sequential input, with the application of the same underlying parameters at each step [2]. There are two types of behavior that can be identified within the RNN: through the regular connections, one layer's outputs are passed *synchronously* to the subsequent layer, while the recurrent connections are *dynamic*, passing data across adjacent time steps asynchronously [2]. In other words, the hidden state H_t of the current time step t is determined by the input of the current time step from the previous layer X_t and the hidden layer of the previous time step H_{t-1} as:

$$H_t = f(X_t W_{xh} + H_{t-1} W_{hh} + b_h),$$

where f is an activation function, W_{xh} and W_{hh} are the weight parameters for the

input from the previous layer and the hidden layer of the previous time step respectively and b_h is the selected additive bias of the hidden layer. Afterwards, the output O_t of the current time step t is calculated, similarly to the multi-layer FNNs, as:

$$O_t = H_t W_{ho} + b_o,$$

where W_{ho} is the weight parameter for the output layer of the current time step and b_o is the selected additive bias of the output layer [2]. Because of the way the recurrent neural networks process information and the fact that they can process input sequences of variable lengths [26], RNNs are often used in speech recognition, time series prediction, natural language processing and, generally, in many tasks with sequential data.

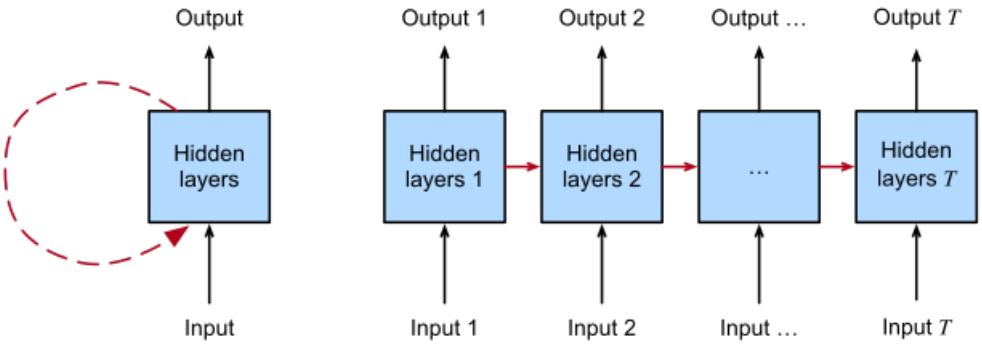


Figure 4.5. *Depiction of a RNN via cyclic edges and unfolded over time steps [2]*

Despite the fact that the basic RNN approach consists of only one hidden layer, the recurrent connections give a sense of depthness in the time dimension [2]. However, stacking hidden layers on top of one another can make a RNN deep in the direction of the input towards the output for each discrete time step, similar to the construction of deep CNNs. Figure 4.6 illustrates a deep RNN with L hidden layers.

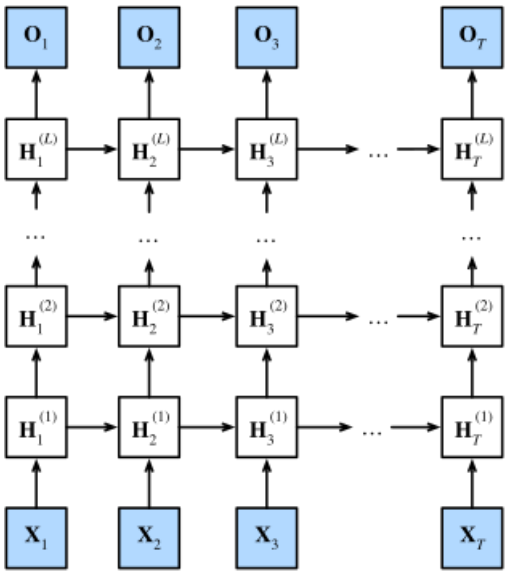


Figure 4.6. *Deep RNN architecture [2]*

4.3.1 Long Short-Term Memory (LSTM)

In theory, a classic RNN, as presented above, can keep track of long-term dependencies of the input sequences. However, during backpropagation, the long-term gradients of the basic RNN can turn to zero, a phenomenon called the *vanishing gradient problem* [75]. In order to address this problem, the *Long Short-Term Memory (LSTM)* model was proposed, replacing the previous recurrent nodes with *memory cells*. Each memory cell contains an internal state, which ensures that the gradient is going to pass through many of the subsequent time steps without vanishing [2], and a number of multiplicative gates that regulate the functions of the LSTM unit.

As shown in Figure 4.7, the main components of a LSTM unit are the forget, input and output gates and the candidate memory. Based on these mechanisms, the hidden state is updated or reset. The values of the three gates are computed via three fully connected layers with *sigmoid* activation functions, in the range of $(0, 1)$, while the candidate memory is calculated in a similar manner by using a *tanh* function with a value range of $(-1, 1)$ [2]. The mathematical equations for these calculations are:

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f),$$

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i),$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o),$$

$$\tilde{C}_t = tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c),$$

where, for each of the three gates and the candidate memory, W_x are the weight parameters for the input, W_h are the weight parameters for the hidden state and b are the additive biases [2]. In order to calculate the internal state of the memory cell C_t at the current time step t, we take into account the internal state at the previous time stepC_{t-1} and the *candidate memory* $\tilde{C}_t$, which contains new data. The *forget gate* F_t dictates how much of the previous internal state C_{t-1} we retain, while the *input gate* I_t determines the importance of the new data, via $\tilde{C}_t$, to the current internal state [2]. This function can be expressed in a mathematical way as:

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t,$$

where $\odot$ is the Hadamard element-wise product operator. Afterwards, the *output gate* O_t determines whether the current internal state C_t should affect the current output (or hidden state) H_t at time step t [2]. This leads to the following equation:

$$H_t = O_t \odot tanh(C_t)$$

Overall, the LSTM has the ability to decide whether the internal state should be adjusted in response to subsequent inputs and whether the hidden state should be impacted by the internal state [2]. This flexibility allows the information to be propagated across many time steps, without affecting the network, and influence a later hidden state, thus

enabling long-term memory effects on the network [2]. Deep LSTM networks are constructed in a similar manner to deep RNNs, employed many applications in time series prediction tasks within the healthcare domain [76].

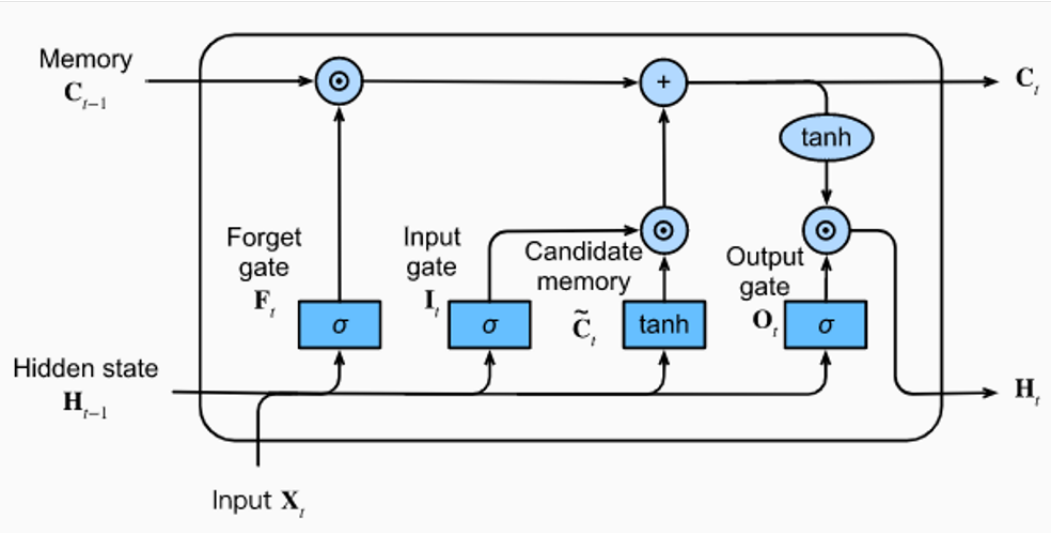


Figure 4.7. *The architecture of a LSTM unit [10]*

4.3.2 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU), proposed by Cho et al. [77], requires fewer parameters than LSTM, while retaining the LSTM memory cell approach. As a result, the less resource-intensive GRU leads to faster computations [78] than LSTM and comparable performance on certain tasks. The main components of a GRU unit are the reset and update gates and a candidate hidden state. Similarly to the LSTM, two fully connected layers with *sigmoid* activation functions, in the range of $(0, 1)$, are employed to compute the values of the two gates. The *reset gate* R_t controls how much of the previous state should be retained, while the *update gate* Z_t controls how much of the old state is copied to the new state [2]. The respective mathematical equations are:

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r),$$

$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z),$$

where, for each of the reset and update gates, W_x are the weight parameters for the input, W_h are the weight parameters for the previous hidden state and b are the additive biases [2]. Similarly to the calculation of the hidden state for a RNN, we compute the *candidate hidden state* $\tilde{H}_t$, with the integration of the reset gate as:

$$\tilde{H}_t = tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h)$$

where, for the candidate hidden state, W_{xh} are the weight parameters for the input, W_{hh} are the weight parameters for the previous hidden state and b_h are the additive biases. Finally, the update gate determines how much each of the previous hidden state

H_{t-1} and the candidate hidden state $\tilde{H}_t$ contribute to the current hidden state H_t [2]. This leads to this final update equation:

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t$$

Overall, the GRU captures short-term dependencies via the reset gate and long-term dependencies via the update gate in the input sequences, thus emulating the behavior of LSTM [2].

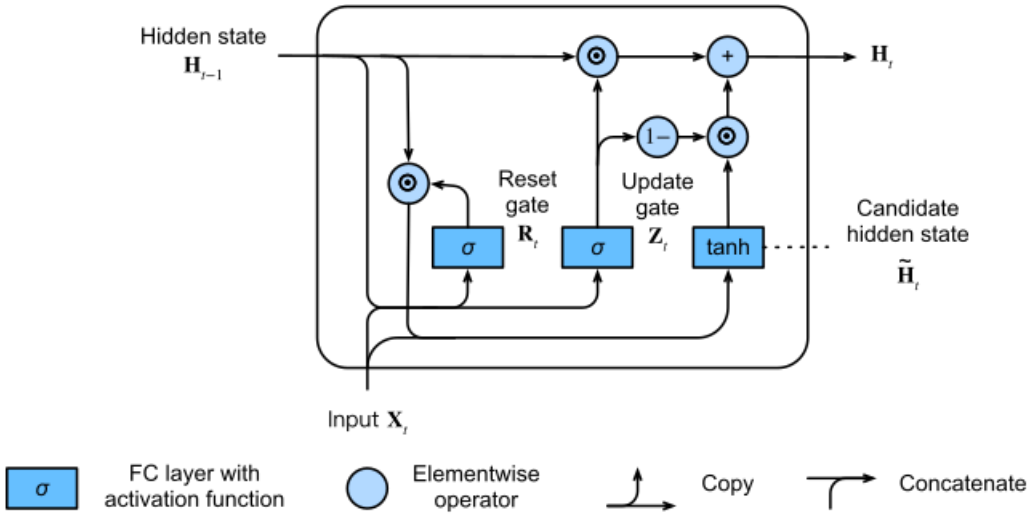


Figure 4.8. *The architecture of a GRU unit [2]*

4.3.3 Bidirectional Recurrent Neural Network (BRNN)

The standard RNN and its variants process the past values of a sequence to make future predictions. However, many tasks require the predictive models to take into account the overall context, both from past and future values, to enhance their performance. On this front, building upon the standard unidirectional RNN structure with another RNN layer, which processes the same input backwards, the *bidirectional recurrent neural network (BRNN)* is implemented [79]. Therefore, the BRNN contains a forward hidden state $\overrightarrow{H_t}$ and a backward hidden state $\overleftarrow{H_t}$, which are computed for the time step t similarly to the standard RNN hidden state, as:

$$\overrightarrow{H_t} = f(X_t W_{xh}^{(\rightarrow)} + \overrightarrow{H}_{t-1} W_{hh}^{(\rightarrow)} + b_h^{(\rightarrow)}),$$
$$\overleftarrow{H_t} = f(X_t W_{xh}^{(\leftarrow)} + \overleftarrow{H}_{t-1} W_{hh}^{(\leftarrow)} + b_h^{(\leftarrow)}),$$

where, for each of the forward and backward hidden states, f is an activation function, W_{xh} and W_{hh} are the weight parameters for the input from the previous layer and the hidden layer of the previous time step respectively and b_h is the selected additive bias of the hidden layer [2]. Next, the forward and backward hidden states are concatenated into a final hidden state H_t, which is passed to the output layer or, in the case of a deep BRNN, to the next RNN layers. Afterwards, the output O_t at the current time step t is

computed as:

$$O_t = H_t W_{ho} + b_o,$$

exactly the same as for a standard unidirectional RNN [2]. In general, bidirectional RNNs are slower models that have high computational cost, thus requiring more time for training. The bidirectional approach is applicable to the other variants (BiLSTM, BiGRU) as well. It is especially useful for natural language processing tasks, such as language translation and sentiment analysis, for time series forecasting and for audio processing tasks, such as speech recognition [80].

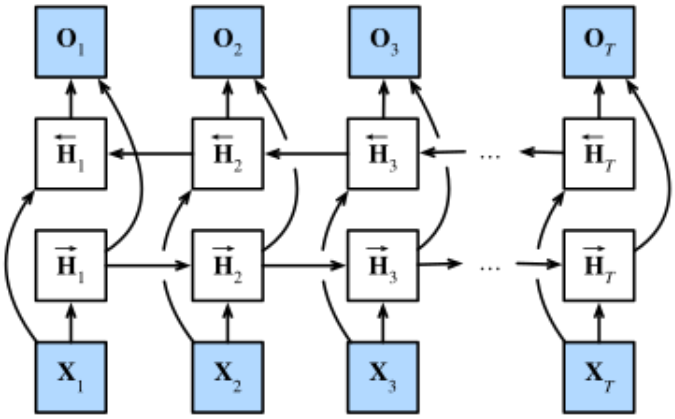


Figure 4.9. *Architecture of a bidirectional RNN [2]*

Chapter 5

Federated Learning

In this chapter, we delve deeper into the concept of federated learning and its implementation. We start by discussing the advantages of federated learning over centralized and distributed ML and presenting the key elements of a typical FL system. We then examine some widely used FL algorithms that aim to optimize different aspects of the FL workflow. Finally, we turn our attention to the privacy and security concerns that arise in FL systems and how mechanisms can be integrated into the FL context to mitigate them.

5.1 Introduction to Federated Learning

5.1.1 Limitations of Centralized Machine Learning

Artificial Intelligence (AI) has been evolving rapidly during these last decades, showing its strengths in numerous industries and in many aspects of everyday life. As a result, researchers and businesses attempt to incorporate ML solutions in continuously more domains, with applications in transportation, healthcare, finance, etc. One of the key factors that drives these tremendous developments in AI is the *Big Data availability* [81]. As it was explained earlier, ML models, in particular, learn to perform specific tasks by training and understanding underlying relationships on available data. Since enormous amounts of data are generated every second, it is expected that ML models will become more accurate, ubiquitous and capable of generalizing. However, the real world scenario does not agree with these expectations. In most domains, the available, centralized data are often limited and of poor quality, impeding the realization of AI technology [81].

The traditional ML workflow is based on the assumption of centralized data training, where the data are gathered and the whole training process takes place at a central server [82]. Nevertheless, there are several challenges to the wide adoption of this approach. In today's world, *data privacy and security* has evolved into a major issue, prohibiting organizations and businesses to access personal data without the users' agreement. Often, this agreement presupposes explicitly informing the users about how their data are going to be handled and utilized, thus complicating the process of collecting their data. For instance, the General Data Protection Regulation (GDPR) [16], enforced by the European Union on May 25, 2018, aims to protect the personal data of EU citizens in such a manner. Recent breaches of data privacy, such as the collection of personal data from Facebook users from the British consulting firm Cambridge Analytica without consent in

the 2010s [83], have led to stricter regulations on data collection and processing, which make their utilization more difficult, even though they are an important step towards data security and transparency. Even within a single organization, data integration between its different sectors is often subject to complex administrative procedures, which further complicate the centralized data gathering [81].

Furthermore, traditional centralized ML algorithms show limitations in terms of *scalability* and *efficiency*, thus being unable to handle large-scale data and model parameters [84]. These limitations translate into the inability of a single computer system to address the high computational and memory requirements of large-scale ML applications. First, as the sizes of the model and the training data grow, the training time becomes increasingly longer, slowing down the development of ML models and the experimentation with different structures. Then, the memory of a training system is often inadequate for fitting a large ML model and batches of data during training, leading to performance degradation or even system failure. A potential solution to this problem is vertical scaling, that is the upgrade of the system's components to improve the memory capacity and processing power, which is expensive and not always sufficient [85]. As a result, it is important to come up with efficient, cost-effective alternatives to centralized machine learning, tackling the aforementioned limitations.

5.1.2 Distributed Machine Learning

One approach, that addresses the latter challenges of large-scale data and model parameters, is *distributed machine learning* (DML), an amalgamation of distributed computing and machine learning [11]. The main idea behind DML is to partition the training data and model to multiple devices, referred as *clients*, which learn each partition as a subtask [11]. A central server is responsible for the model partitioning and instructing the data manager, which could be the server or a third-party storage system, on data partitioning [11]. As each client works on their allocated subtasks, the training process is parallelized. The clients could work independently or communicate with each other, if a specific subtask depends on the output of another [11]. Once all clients have completed their work, a central server aggregates the clients' models into a final, complete model [11]. Figure 5.1 shows a simple DML system, in accordance with the workflow described above.

However, the DML framework is also accompanied with its own basic assumptions. In order to design the data partitioning scheme, the central server needs to have access to the whole dataset [11]. Otherwise, if the clients generate the data locally, it is assumed that the local datasets are independent and identically distributed (IID) and have roughly the same size [86]. This powerful assumption on data homogeneity significantly restricts the applications of DML, as the local datasets can be subject to privacy regulations and often showcase certain levels of heterogeneity. Then, DML relies heavily on communication between the central server, the data manager and the participating clients, causing high communication costs, slow training times and increased vulnerability to malicious attacks. Consequently, the participants are often computing nodes within a data cen-

ter, with powerful computational capabilities, supported by a fast and reliable network infrastructure [28]. Overall, distributed machine learning mainly aims at the horizontal scaling of the training system, while it shows certain limitations related to data privacy and security and the system requirements.

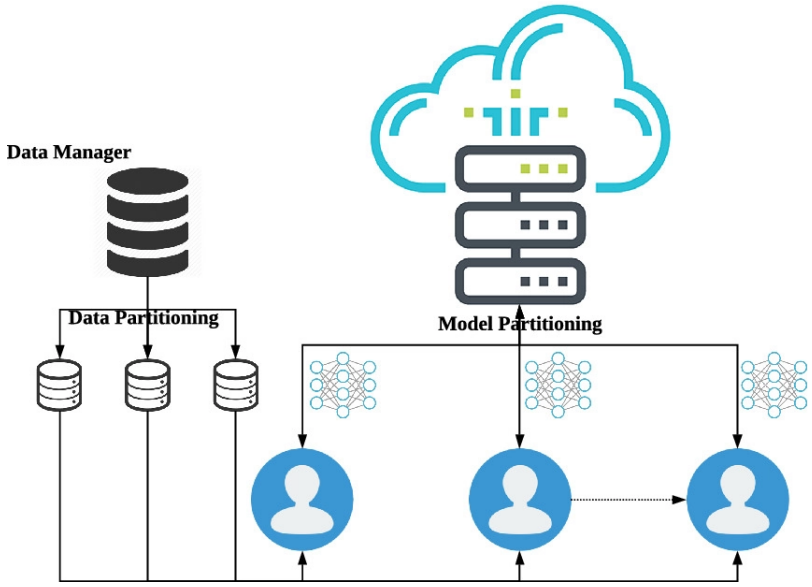


Figure 5.1. *A basic distributed machine learning system [11]*

5.1.3 The Concept of Federated Learning

Federated learning (FL) is a term first introduced by McMahan et al. [27], to describe a machine learning framework where "the learning task is solved by a loose federation of participating devices (which we refer to as *clients*) which are coordinated by a central *server*" [27]. This approach aims at reducing the data privacy and security risks deriving from traditional, centralized ML, as multiple clients collaboratively train a model while keeping the training data decentralized [28]. Only the locally-computed client updates on a global model are communicated via the network, which are subsequently aggregated by the central server to perform an update on the global model [27].

The underlying assumptions of federated learning are what differentiates it from distributed machine learning. One of the main properties of federated learning is that the local datasets are not required to be independent and identically distributed. Far from it, the local datasets are typically non-IID, thus not representing the population distribution, and vary in size [27]. This enables the application of federated learning in domains where local dataset heterogeneity is expected, such as in finance, healthcare, IoT and telecommunications. For instance, federated learning is a potential solution for the collaboration of medical centers with privacy-sensitive data on various tasks, such as the early ICU mortality risk prediction that is studied in this thesis. Then, depending on the federated setting, the requirements on the participating devices are looser than in DML. In the cross-device federated setting, the number of participants is expected to be significantly larger than the average size of the local datasets [27], while slow or non-responsive clients

do not cause the FL training process to be aborted. Consequently, federated learning addresses many of the limitations of centralized ML and distributed ML, thus representing a promising approach for training ML models on decentralized data in a privacy-preserving manner.

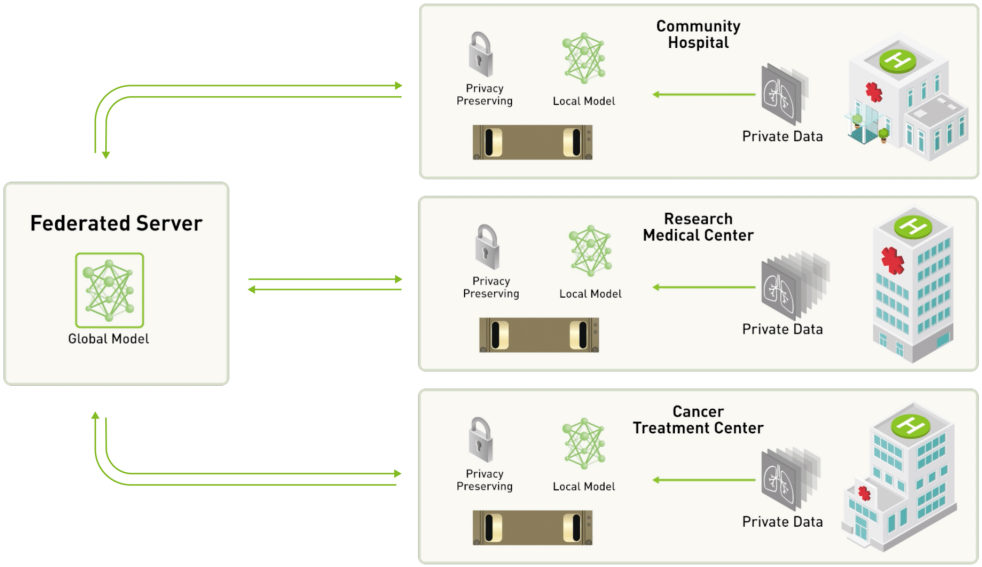


Figure 5.2. *A federated learning system in healthcare, with a federated server and a privacy-preserving mechanism [3]*

In this work, we focus on the non-IID and unbalanced properties of federated optimization [27] within the healthcare domain. We recreate a horizontal federated learning scenario, where the participants share the same feature space, but their local datasets contain different samples [81]. In our setting, this corresponds to a group of medical centers that monitor the same health indicators for their own ICU patients.

5.2 Federated Learning Workflow

The basic federated learning setting consists of a centralized server, orchestrating the training process, and the participating nodes, which locally compute updates to the global model [28]. The objective function for federated learning is the weighted average of the local objective functions of each participating client. In a mathematical manner, the objective function can be expressed as:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w),$$

where w are the model weights, K is the number of participating clients, n is the total number of samples in the local datasets of the K clients, n_k is the size of the local dataset of client k and F_k is the local objective function of client k [27]. The goal is to minimize this objective function, that is to find a set of weight updates to a global model from the clients that lead to the optimal global model parameters.

Before the learning process commences, the central server decides on the most suitable machine learning model for the FL task at hand. This model can be initialized either with a random set of weights or with the weights of a pre-trained model on this task. At the same time, if needed, the clients are instructed to collect and store their training data locally [28], structured in a specific way for training the ML model. Each iteration of the learning process constitutes a *FL round*, which leads to an update to the global model by the central server. A typical FL round consists of the following steps:

1. **Client Selection:** The central server selects a subset of the clients to participate in the subsequent round of federated learning, if they meet certain eligibility requirements [28]. For instance, mobile devices could be required to be fully charged and connected to a stable, fast Wi-Fi network, in order to participate in the training process.

2. **Broadcast:** The server broadcasts the current global model parameters and the chosen training method to the selected cohort of clients [28].

3. **Client Computations:** Each client computes an update to the current global model on their local dataset, following the training algorithm that was predefined by the central server [28].

4. **Aggregation:** Once the clients have computed their model updates, they are communicated and aggregated to the server. A privacy-preserving mechanism may be incorporated in this stage, such as *differential privacy* or *secure aggregation*, as an additional security measure [28]. The training round is considered successful if a sufficient number of clients reports their local model updates on schedule. Otherwise, it is aborted and a new FL round commences [87].

5. **Model Update:** Afterwards, the global model, maintained by the server, is updated based on the aggregated client update, computed at the current round [28]. If the termination criterion is met (a certain number of training rounds is reached or an early stopping mechanism is activated), the server updates all the clients with the final model and signals the end of the training process [87]. Otherwise, the server broadcasts the latest model parameters to a new cohort of clients and the next FL round commences.

The model updates, in the procedure described above, are considered synchronous. However, asynchronous federated learning approaches have been proposed as well. One such approach is asynchronous SGD, where each client's local update is applied to the global model, once it is computed, without aggregation with other client updates [28]. Additionally, *split learning* techniques for deep learning have been suggested, where clients train certain layers of deep neural networks and communicate them to the server, which completes the model training with forward propagation, without access to the raw data [88]. Another assumption of basic FL strategies is that the local models share the same global model architecture. Aiming to address client heterogeneity, the HeteroFL framework was proposed in 2020, which enables the training of heterogeneous models from

clients with varying computational and communication capabilities to produce a single global inference model [89].

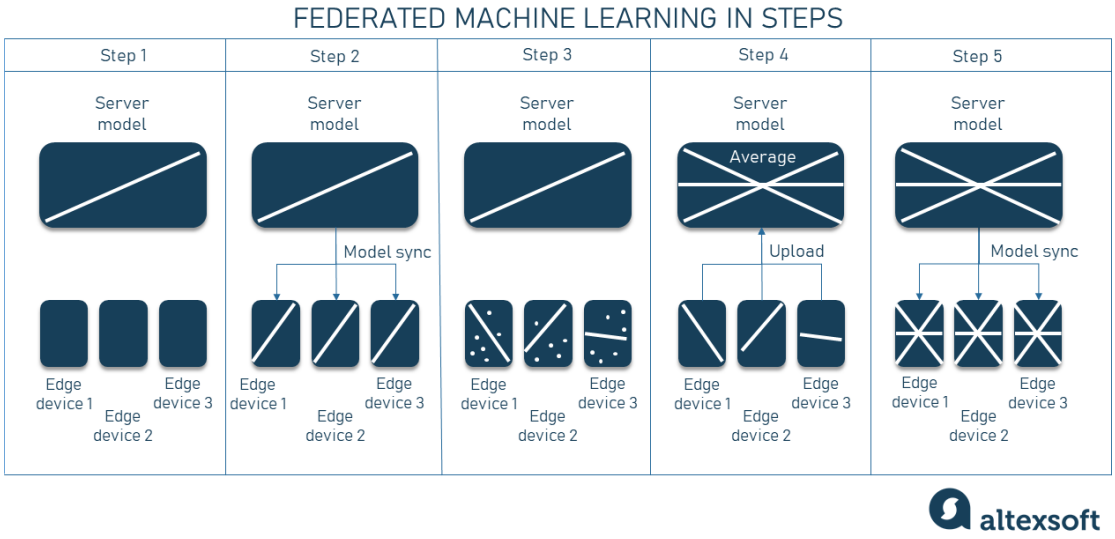


Figure 5.3. *Federated machine learning process in steps, altexsoft [12]*

Then, certain decentralized FL settings eliminate the need of a central server, to coordinate the learning process (Figure 5.4), thus preventing single point system failures. For instance, a blockchain-based FL setting for autonomous vehicles was proposed by Pokhrel et al. [90], which integrates federated learning with blockchain technology. However, the network topologies to support this approach may affect the overall performance, by increasing the communication costs [28]. Also, a central authority may still be required to set up the learning process, deciding on the model, the training algorithm, the hyperparameters, etc. [28]

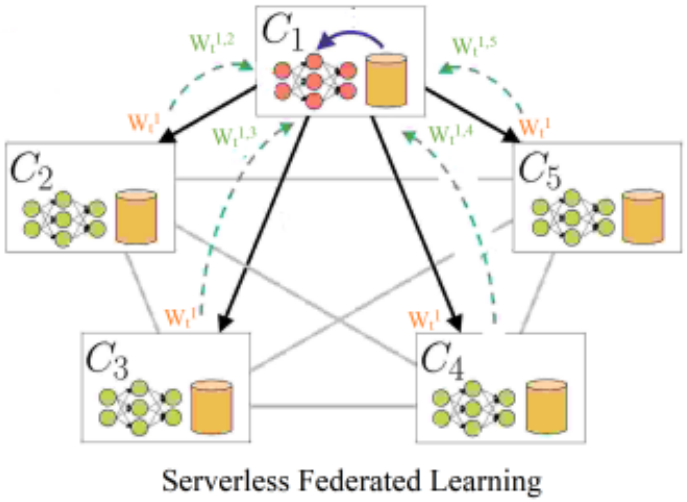


Figure 5.4. *A decentralized federated setting, without the orchestration of a central server [13]*

5.3 Federated Learning Algorithms

The selection of the FL training algorithm is paramount to the performance of the final global model. Heterogeneous federated learning environments present challenges that need to be addressed by employing effective, federated strategies. Different algorithms aim to optimize different aspects of the FL workflow, such as the design of the global and local objective functions, server and client optimization, etc. In this section, we present some state-of-the-art federated learning algorithms, some of which are implemented in the stage of experimental analysis of this thesis.

5.3.1 Federated Stochastic Gradient Descent (FedSGD)

In Chapter 4, the central role of gradient descent, as an optimization method in deep learning training, was highlighted. The gradient descent algorithm iteratively adjusts the model's parameters using the gradients of the loss function on the whole training set, as:

$$w := w - \eta \nabla Q(w) = w - \frac{\eta}{n} \sum_{i=1}^{n} \nabla Q_i(w),$$

where w are the model's weights, η is the learning rate, n is the number of data points in the training set, Q_i is the value of the loss function on the i-th data point and Q the value of the loss function on the training set. Gradient descent moves in the direction of the steepest descent to minimize prediction error. However, calculating a step of the gradient descent on the entire training set can be computationally expensive, slowing down the learning process. *Stochastic gradient descent* (SGD) addresses this challenge by computing gradients on a random subset of the training dataset at each iteration. Typically this subset contains more than 1 data point, to achieve smoother convergence and reduced variance in the weight update and to enable the use of highly optimized matrix operations (thus referred as *mini-batch*) [91].

Federated stochastic gradient descent (FedSGD) [29] is a basic adaptation of the SGD algorithm to the federated setting. A federated environment contains another level of separation among the data, as they are split into multiple local datasets. As a result, on each round, a random fraction C of the FL clients is selected to participate in training, where the gradient of the loss function is computed on the entire local dataset of each of these clients [27]. The gradients are averaged, according to the number of training data per client, to produce the next gradient descent step. This can expressed with the following mathematical equation:

$$w_{t+1} := w_t - \eta \nabla F(w_t) = w_t - \eta \sum_{i \in K} \frac{n_i}{n} \nabla F_i(w_t),$$

where K is the subset of clients that participate in the FL training round. Parameter C determines the global mini-batch size, as $C = 1$ means all the clients participate in the FL round (corresponding to the full-batch gradient descent algorithm) [27], and $C = \frac{1}{n}$ means only one random client participates in the FL round (corresponding to the SGD algorithm).

5.3.2 Federated Averaging (FedAvg)

Federated Averaging (FedAvg), introduced by McMahan et al. [27], is a generalization of FedSGD and constitutes one of the most widely used federated learning algorithms. FedAvg proposes that the server takes a weighted average of the client model weights, rather than the computed gradients, to update the global model. This approach was developed under the premise that, if all clients start with the same model, averaging the gradients is equivalent to averaging the model weights themselves [27]. Consequently, this model weight update can be expressed as:

$$w_{t+1} = \sum_{i \in K} \frac{n_i}{n} w_{t+1}^i.$$

Respective of how the FedAvg algorithm is formulated, each client may perform more than one training round on their entire local dataset, before the model averaging phase [27]. Therefore, the number of local training epochs E and the local mini-batch size B are two additional parameters that need to be set before the FL process commences, along with the fraction of participating clients C [27].

In homogeneous federated scenarios, where all local datasets are independent and identically distributed (IID), the FedAvg algorithm performs considerably well. However, its efficiency is not guaranteed in the presence of data heterogeneity, with non-IID local datasets. The final global model may become biased towards the datasets of certain FL participants, unable to generalize sufficiently to unseen data.

5.3.3 Federated Optimization in Heterogeneous Networks (FedProx)

A federated learning setting may present two types of heterogeneity. *Systems heterogeneity* refers to the differences, in terms of system characteristics, on each participating node [30]. Typically, devices with limited computational and communication capabilities may be ignored during FL training, if they are straggling to complete their tasks [87]. However, by excluding these clients from the learning process due to their technical limitations, one may negatively affect convergence and induce bias in the final model [30]. *Statistical heterogeneity* refers to non-IID, unbalanced local datasets and poses a challenge for the generalizability of the final model, as it was discussed above.

To address the heterogeneity in a FL environment, Li et al. [30] proposed *FedProx*, a FL framework that can be seen as a generalization and re-parametrization of FedAvg. This algorithm aims to correct *client drift*, a drift of local updates that causes slower convergence, due to heterogeneity. In order to do so, FedProx introduces a *proximal term* to the client tasks, which restricts the local updates to be closer to the latest global model update [30]. Instead of minimizing their local loss function $F_i(w)$, a client i approximately minimizes a new objective function h_i:

$$h_i(w; w^t) = F_i(w) + \frac{\mu}{2}\|w - w^t\|^2,$$

where μ is the proximal term, w are the variable weights of the local model during

training and w^t are the weights of the latest global model. The use of the proximal term deals with the issue of statistical heterogeneity with the restrictions on local updates and enables the efficient aggregation of variable amounts of local training, by removing the need to manually set the number of local training epochs beforehand [30].

A similar approach, addressing the phenomenon of client drift, is *SCAFFOLD* [92]. It measures the drift in the local updates of each client, during every FL round, and adjusts their local updates accordingly. Client drift can be expressed as the difference between the global model update and the client's local update. Overall, both FedProx and SCAFFOLD are FL approaches that focus on the optimization of local training, to address slow convergence and generalizability issues.

5.3.4 Adaptive Federated Optimization (FedOpt)

Another approach, addressing the undesirable convergence behavior of FedAvg in heterogeneous settings, is to experiment with different optimization methods both on client and server level. The server update of FedAvg is equivalent to applying SGD optimization with learning rate $\eta = 1$ to the aggregated "pseudo-gradient" [31]. However, in non-federated learning environments, adaptive optimization techniques have demonstrated considerable performance in tackling convergence issues. This motivated Reddi et al. [31] to propose an adjustment of adaptive optimizers in a federated setting. They refer to the family of algorithms, that employ different optimization techniques than SGD for client optimization or the model update on the server, as *FedOpt*. By maintaining SGD for client optimization and using adaptive optimization methods on the server-side, they specialize FedOpt from a global perspective. Their proposed algorithms are *FedAdagrad, FedAdam and FedYogi*, which employ the Adagrad, Adam and Yogi optimizers in the server optimization step respectively. The optimization step for each of these algorithms is expressed as:

$$u_t = u_{t-1} + \Delta_t^2 \qquad \textbf{(FedAdagrad)}$$

$$u_t = \beta_2 u_{t-1} - (1-\beta_2)\Delta_t^2 \qquad \textbf{(FedAdam)}$$

$$u_t = u_{t-1} - (1-\beta_2)\Delta_t^2 sign(u_{t-1} - \Delta_t^2) \qquad \textbf{(FedYogi)}$$

where β_2 is a decay parameter and Δ_t is the weighted average of the local gradients, obtained from each client. The model weights are updated based on the following mathematical expression:

$$x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{u_t} + \tau},$$

where η is the learning rate and τ is referred as *degree of adaptivity*. Variable m_t has been previously updated based on the current "pseudo-gradient" and another decay parameter β_1, according to the equation $m_t = \beta_1 m_{t-1} + (1-\beta_1)\Delta_t$ [31]. Since these

FL algorithms concentrate on server-side optimization, they can be integrated with the previous algorithms that focus on client-side optimization (FedProx, SCAFFOLD) for better results. Overall, FedAdagrad, FedAdam and FedYogi are promising FL algorithms that enhance the efficiency and performance of federated learning, especially in complex, non-stationary data environments.

5.3.5 Federated Averaging with Server Momentum (FedAvgM)

Following the direction of the previous FedOpt algorithms, Hsu et al. [32, 33] propose the use of adaptive server optimization in the FL setting by adding momentum at the server, with their FL algorithm *FedAvgM*. It has been proven that momentum on top of SGD leads to faster convergence by a running accumulation of gradient history to dampen oscillations. This behavior seems applicable to heterogeneous FL settings, where participants have a sparse distribution of data and a limited number of samples [32]. Instead of updating the model weights with the weighted average of the local updates in the basic version of FedAvg, a variable u is calculated:

$$u = \beta u + \Delta w,$$

where β is the momentum parameter and Δw is the weighted average of the local model weights. Then, the global model weights are simply updated according to u:

$$w = w - u.$$

By testing FedAvgM on FL scenarios with increasingly non-IID data, it was shown that FedAvgM maintains a relatively good performance even in highly heterogeneous settings. On the same FL scenarios, the performance of FedAvg deteriorated significantly, as the datasets became progressively more non-identical [32].

5.4 Privacy and Security in Federated Learning

One of the main advantages of federated learning is the improvement of data privacy and security over centralized and distributed machine learning. Gathering the raw client data to a central server creates an attractive attacking point for potential adversaries, since they may gain access to the entire dataset. The FL framework keeps the data decentralized, making client devices the only breach point for accessing raw data. Even if these adversaries hack a client device, they acquire only a small, single-sourced subset of the entire dataset.

Nevertheless, federated learning still has certain vulnerabilities and security risks that prevent it from guaranteeing privacy and security [11]. The local client updates can be targeted for individual inference attacks, while the global model parameters provide an insight into the characteristics of the entire dataset. Most data leaks occur when data are in transit, highlighting the need for secure communication. This has motivated the development of encryption mechanisms, which ensure that any malicious third party would

be unable to make any inferences from the ciphered version of the data. The federated learning framework can be integrated with modern privacy and security algorithms, which further strengthen it as a privacy-preserving alternative to other ML frameworks.

5.4.1 Differential Privacy

Differential privacy (DP) is a privacy-preserving concept, built upon the premise that, if a substitution of single item does not significantly affect the statistical properties of the database, then joining the database does not expose any individuals to large inference attacks [93, 94]. Regarding centralized DP, a randomized algorithm M gives *ϵ-differential privacy* for any pair of neighbouring datasets D and D' (which differ on a single element) and every set of outcomes Ω when:

$$Pr[M(D) \in \Omega] \leq \exp^{\epsilon} \cdot Pr[M(D') \in \Omega],$$

where ϵ is the parameter that controls the level of privacy preservation [11]. As ϵ grows smaller, the criteria for differential privacy are stricter, thus guaranteeing a higher level of privacy. Randomized algorithms for DP add noise to the dataset from a predetermined distribution, leading to the widely used *Laplacian* and *Gaussian* mechanisms for the respective distributions. However, in many instances, the individuals require further protection, even against the data manager that gathers their data. As a result, an improved model, referred as *ϵ-local differential privacy*, was proposed, preventing the data manager to learn too much sensitive information about any individual data contributors [11]. The algorithm is similar to the one presented above, for any two inputs x and x' and any possible output Ω:

$$Pr[M(x) = \Omega] \leq \exp^{\epsilon} \cdot Pr[M(x') = \Omega].$$

Local DP adds randomized noise over each user's data separately, in contrast to the standard DP, where randomized noise, deriving from the same algorithm, is added to the combined dataset [11]. As a result, local DP is considered a stronger privacy-guaranteeing mechanism, since the data collector is unable to distinguish between the original data x and the noise-distorted data x'.

These differential privacy models are often incorporated into deep learning, because of their desirable properties of composability, group privacy and robustness to auxiliary information[95]. McMahan et al. [27] were the first to suggest using DP to protect client data in the federated learning setting. In the FL setting, the standard DP model works on the server-side, treating the client updates as a "dataset", where each client update is a "data instance". The goal of DP in this scenario is to "hide" the individual client updates during the model aggregation process [11]. Setting the sensitivity of DP is crucial for improving privacy without deteriorating the global model's performance. However, this procedure may prove to be quite challenging, since the server should not be able to distinguish individual client updates [11]. In a similar manner, local DP can be adapted into the FL setting, with clients sending noisy local updates to the server for

aggregation. Even though it provides greater privacy, the total volume of added noise to the local updates is significantly higher than in standard DP, which can negatively affect performance. Overall, there is a trade-off between privacy and performance with the integration of DP into the FL setting, which requires further exploration in the future.

5.4.2 Secure Aggregation

Security mechanisms are typically employed to provide secure data transmission over a communication channel, with the use of cryptography techniques and protocols. In the FL setting, the phase of model aggregation involves the largest amounts of communication, since the participants send their local model updates to the central server for averaging. To better protect individual model updates, the *secure aggregation* protocol encrypts each client update before their transmission over the network. Secure aggregation guarantees that any third party, including the central server, may have access only to the encrypted version of local updates [11]. Some widely used security protocols are secret sharing schemes, homomorphic encryption and secure multiparty computation, the family of security algorithms where secure aggregations belongs into.

Yao et al. [96] was the first to propose the concept of *secure multiparty computation*. The underlying problem is that a set of parties with private inputs collaborate to compute an agreed-on function over their data, without compromising their privacy. The idea of secure multiparty computation is that an encryption scheme can provide this privacy guarantee without diminishing the utilization of the data. These algorithms seem directly applicable to federated learning, where client updates, containing sensitive information, are aggregated to a server and both privacy and model performance are of utmost importance [11].

Secure aggregation is a subclass of secure multiparty computation algorithms, where a group of trustless parties with sensitive information cooperate in calculating an aggregated value [11]. Bonawitz et al. [97, 98] proposed the first secure aggregation protocol for federated learning, where the server learns only the sum of clients' model updates. This initial approach later developed into a full version for practical applications, where each local client update is masked by a random number, preventing the server from accessing it directly, and each client generates a private-public key pair for each FL round and a private shared key with a hash function, by combining their private key with the public keys of the other participants. The hash function guarantees that each pair of private shared keys is additive inverse, thus allowing the server to effectively offset the effect of the masks during aggregation to calculate an accurate result [11]. However, client disconnections and reconnections at various moments can prevent the server from offsetting the effect of the masks, a problem that can be tackled with additional properties, such as a second mask. Overall, this protocol provides a comprehensive and robust security guarantee over the model aggregation process [11].

Chapter 6

Related Work

Mortality risk prediction tasks hold paramount importance in the healthcare domain, as they assist in identifying patients who are at high risk and require immediate attention. As such, predicting the probability of a patient dying during hospitalization is one of the most researched clinical prediction tasks [99]. The need for developing effective models that address this challenge is even more prominent in the ICU setting, where patients with a severe health status and critical medical conditions are admitted and treated. Regarding this issue, early warning and scoring systems, like APACHE [34] and SAPS [35], have been utilized to estimate patient mortality, since they monitor various clinical variables, such as vital signs and lab results, during the first hours of an ICU stay. However, the widespread adoption of electronic health records (EHRs), leading to vast amounts of patient data with rich temporal dynamics being generated rapidly, has motivated researchers to seek other techniques to utilize the available information better for early mortality prediction.

6.1 Centralized ML Approaches

Machine learning algorithms can learn complex relationships among clinical variables and patient outcomes that conventional models may be unable to capture, and develop effective, data-driven models that make accurate estimations of patients' mortality risk. Johnson and Mark [100] focus on real-time mortality prediction in the ICU, developing Logistic Regression (LR) and Gradient Boosting (GB) models having both dynamic features, such as physiologic and laboratory measurements, and static features, such as gender, age, etc., as their input. Their results on the MIMIC-III database [36] showed that the AUROC score of a GB model (0.920) was considerably higher than the AUROC of the conventional SAPS-II scoring system (0.809). Working on the MIMIC-III database as well, Purushotham et al. [101] presented a benchmark of deep learning models on various medical tasks, including mortality prediction, and showed that they consistently outperform both traditional machine learning models and conventional scoring systems, especially when the input data were not pre-processed.

Subsequently, Awad et al. [102] proposed a framework for early mortality prediction in the ICU and conducted a thorough time-series analysis on the performance of different mining methods, namely random forest (RF) classifiers, partial decision trees (PART) and Bayesian Networks (BN) algorithms, during the first 48 hours of an ICU stay. Their results

on the MIMIC-II database [103] showed that ML classifiers, with data from the first 6 hours of ICU stays, outperformed conventional scoring systems, with access to data from the first 48 hours of ICU stays, while their best performing model was a RF classifier on 48 hours after ICU admission with an AUROC of 0.83. Additionally, Pattalung et al. [104] proposed a data-driven framework for ICU mortality prediction with time-series, combining high-accuracy RNNs with the SHAP system [105] that calculates the contribution of each variable for a prediction, thus providing an interpretable explanation. Their experiments were conducted on three public critical care databases (MIMIC-III, MIMIC-IV [106] and eICU [17]) with their models achieving an AUROC of 0.87-0.91.

6.2 Federated Learning Approaches

With regard to multi-center collaboration and patient data privacy, several federated learning solutions have been proposed, however, focusing on *in-hospital mortality prediction* rather than on the ICU setting. Lee and Shin [107] experimented on the MIMIC-III database by setting up a federated environment with 3 clients and evaluating the performance of a LSTM model. Their results indicated that the FL model can reach a comparable performance to the CML model, in terms of AUROC and F1-Score, while they observed that the performance of FL with imbalanced client datasets (regarding size, not distribution) was scarcely affected. Budrionis et al. [108] extended on the work of Purushotham et al. [101] on the MIMIC-III database, studying different parameters that may affect a federated learning environment. Their experiments include increasing the amount of data in the system with a constant number of FL clients, increasing the number of FL clients with a constant amount of data and using imbalanced local datasets (as with [107], only in terms of size). Their predictive models performed better as the training data increased, while their performance remained largely unaffected by an increase in the number of FL clients or different local dataset sizes.

At the same time, many federated learning solutions shift their focus to other clinical variables and setups to address the mortality prediction task. Huang et al. [109] proposed *community-based federated learning* (CBFL), a novel method that clusters EMR data into several clinically meaningful communities training their own FL models. The clustering criteria and training features were the drugs administered to the patients during the first 48 hours of the ICU stays. Besides training, the clustering process is also incorporated into the FL framework. Their results in different FL scenarios were comparable to CML, in terms of AUROC, both for ICU mortality prediction and for length of ICU stay prediction, another crucial medical task. Moreover, motivated by the recent COVID-19 outbreak, Vaid et al. [110] employed federated learning, utilizing real-world data from 5 hospitals to address 7-day mortality prediction for hospitalized COVID-19 patients. They developed LASSO and multi-layer perceptron (MLP) models and compared their performance with various ML approaches. Their results showed that the FL models outperformed their locally-developed counterparts, enhancing the view that federated learning is a promising privacy-preserving alternative to traditional ML approaches within healthcare.

6.3 Recent Relevant FL Studies

Regarding more recent studies, Dang et al. [14] experimented on the real world, multi-center eICU database and evaluated the performance of several state-of-the-art FL algorithms on two significant medical tasks, in-hospital mortality and AKI prediction. Their results on the in-hospital mortality prediction task, shown in Figure 6.1, indicate that the performance of most FL algorithms was comparable to CML, in terms of AUROC and AUPRC, despite the existence of statistical heterogeneity across participants. Their features were static, obtained at the 24-hour mark after ICU admission, and consisted of basic demographics and statistics of several measurements up to that point, while their predictive model was a simple neural network with 2 fully-hidden layers. It is important to mention that the authors propose the evaluation of different FL methods on time series data and more complex model architectures (which are the focus of this work) in healthcare, as a future research direction.

Method	AUC-ROC (95% CI)	AUC-PR (95% CI)
Local	0.833 (0.808–0.859)	0.472 (0.373–0.480)
Centralized	0.918 (0.907–0.929)	0.668 (0.633–0.701)
IIL	0.877 (0.857–0.897)	0.505 (0.451–0.559)
CIIL	0.828 (0.803–0.852)	0.426 (0.373–0.480)
FedAvg	0.901 (0.882–0.921)	0.638 (0.584–0.688)
FedProx	0.895 (0.877–0.914)	0.577 (0.523–0.630)
FedAvgM	0.906 (0.888–0.925)	0.645 (0.591–0.695)
FedAdam	0.890 (0.870–0.911)	0.578 (0.524–0.631)
FedAdagrad	0.893 (0.873–0.913)	0.596 (0.543–0.649)
FedYogi	0.895 (0.875–0.915)	0.594 (0.539–0.646)

Figure 6.1. *Global Test Performance on the In-Hospital Mortality Prediction Task, Dang et al. [14]*

Furthermore, giving more emphasis to the ICU setting, Mondrejevski et al. [37] proposed FLICU, a workflow for analyzing ICU mortality with MTS data by integrating sequential deep neural network models into the FL framework, using the MIMIC-III database. In terms of data pre-processing and model architecture, they follow the approach described in Pattalung et al. [104]. They evaluated their models (CNN, RNN, LSTM and GRU) with a varying number of FL participants (2, 4 and 8 clients) and with different time windows of vital signs and lab results before ICU discharge or death (8-hour, 16-hour, 24-hour, and 48-hour windows). Their FL approach, using the standard FedAvg algorithm, showed comparable performance to the ideal CML approach and outperformed the privacy-preserving, but inefficient, LML approach, unaffected by the number of FL participants. However, in their study, they assume that the moment of discharge or death

is known beforehand, as they use data from the end of the ICU stay that do not allow early prediction. Thus, the retrospective nature of their analysis prevents the use of their workflow for clinical decision-support.

In response to these limitations, Randl et al. [15] built upon their research, focusing on predicting ICU mortality at an early stage with MTS data from the beginning of the ICU stay. They adjust the workflow of Mondrejevski et al. [37] to the vital signs and laboratory results from the beginning of the ICU and evaluate their GRU-based models on analogous scenarios. They also suggest that F1-Score is a more suitable early stopping criterion for training, as it is the harmonic mean between precision and recall and shows a more defined maximum during training, compared to minimum loss. Figure 6.2 shows their results, indicating that the performance of FL is comparable to CML and considerably better than LML, especially as the number of participating clients increases. However, both works [37, 15] show certain limitations. The MIMIC-III database consists of data from a single medical center, which have been allocated to the simulated FL clients horizontally with stratified splits. Therefore, the federated environment is largely homogeneous, both in terms of data distribution and patient characteristics, which does not correspond to real-world cases.

score	CML	FL			LML		
		2 clients	4 clients	8 clients	2 clients	4 clients	8 clients
$\Delta t_{data} = 8\,h$; avg. $\Delta t_{pred} = 35.2\,h$							
AUROC	0.87 ± 0.01	0.87 ± 0.01	0.87 ± 0.01	0.87 ± 0.01	0.86 ± 0.01	0.84 ± 0.01	0.80 ± 0.01
AUPRC	0.39 ± 0.04	0.38 ± 0.04	0.37 ± 0.04	0.37 ± 0.03	0.36 ± 0.05	0.32 ± 0.04	0.28 ± 0.02
F1	0.40 ± 0.03	0.38 ± 0.02	0.39 ± 0.04	0.39 ± 0.04	0.39 ± 0.03	0.36 ± 0.03	0.35 ± 0.02
precision	0.42 ± 0.07	0.38 ± 0.05	0.35 ± 0.06	0.35 ± 0.08	0.37 ± 0.05	0.37 ± 0.05	0.35 ± 0.04
recall	0.41 ± 0.08	0.41 ± 0.11	0.46 ± 0.08	0.47 ± 0.09	0.43 ± 0.04	0.36 ± 0.03	0.34 ± 0.03
$\Delta t_{data} = 16\,h$; avg. $\Delta t_{pred} = 27.2\,h$							
AUROC	0.89 ± 0.01	0.88 ± 0.01	0.88 ± 0.01	0.88 ± 0.01	0.87 ± 0.01	0.85 ± 0.01	0.82 ± 0.01
AUPRC	0.44 ± 0.05	0.41 ± 0.04	0.41 ± 0.04	0.42 ± 0.03	0.40 ± 0.05	0.37 ± 0.04	0.33 ± 0.03
F1	0.44 ± 0.02	0.41 ± 0.01	0.40 ± 0.05	0.43 ± 0.03	0.41 ± 0.02	0.39 ± 0.03	0.37 ± 0.02
precision	0.46 ± 0.05	0.41 ± 0.04	0.43 ± 0.06	0.43 ± 0.04	0.38 ± 0.08	0.38 ± 0.05	0.41 ± 0.03
recall	0.42 ± 0.05	0.42 ± 0.07	0.40 ± 0.11	0.43 ± 0.06	0.46 ± 0.08	0.42 ± 0.06	0.33 ± 0.04
$\Delta t_{data} = 24\,h$; avg. $\Delta t_{pred} = 19.2\,h$							
AUROC	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.00	0.89 ± 0.01	0.89 ± 0.01	0.87 ± 0.01	0.83 ± 0.01
AUPRC	0.50 ± 0.04	0.49 ± 0.04	0.47 ± 0.04	0.47 ± 0.03	0.47 ± 0.04	0.43 ± 0.04	0.37 ± 0.03
F1	0.49 ± 0.04	0.48 ± 0.03	0.46 ± 0.03	0.44 ± 0.03	0.45 ± 0.02	0.42 ± 0.02	0.40 ± 0.02
precision	0.55 ± 0.05	0.52 ± 0.05	0.49 ± 0.07	0.45 ± 0.07	0.48 ± 0.08	0.39 ± 0.04	0.46 ± 0.02
recall	0.46 ± 0.08	0.46 ± 0.05	0.44 ± 0.05	0.46 ± 0.11	0.46 ± 0.10	0.48 ± 0.09	0.35 ± 0.03

Figure 6.2. *Results of Randl et al. [15] with F1-Score as early stopping criterion*

In this thesis, we address the aforementioned limitations by using the real world, multi-center eICU database. It allows us to use actual hospitals as FL clients, thus recreating a realistic, heterogeneous federated environment to conduct our experiments. Since our FL scenarios involve non-IID datasets, we do not limit ourselves to the standard FedAvg algorithm, but evaluate other federated learning algorithms that address the challenges of heterogeneous environments, similar to Dang et al. [14]. Moreover, we built upon Randl et al. [15], evaluating more RNN model architectures on the early ICU mortality risk prediction and their incorporation into the federated learning framework.

Chapter 7

Data & Preprocessing

In this chapter, we focus on presenting the dataset utilized in the study. It begins by introducing the eICU Collaborative Research Database and presenting some key information about its contents. Next, the cohort selection criteria are discussed and the cohort demographics are displayed. Following the cohort selection, the chapter delves into the data preparation process that precedes ML training and evaluation.

7.1 Dataset

For the purposes of this thesis, we utilize the *eICU Collaborative Research Database* [17], managed by the MIT Laboratory for Computational Physiology. The eICU is populated with real-world data from multiple critical care units throughout the United States, participating in the Philips eICU telehealth program. The multi-center nature of the eICU database makes it an ideal choice for recreating a realistic, heterogeneous federated learning environment. It contains over 200 thousand patient intensive care unit (ICU) stays for over 139 thousand unique patients, admitted to 208 different hospitals between 2014 and 2015. In order to meet the safe harbor provision of the US Health Insurance Portability and Accountability Act (HIPPA), each database table is de-identified and all protected health information is removed, in addition to hospital and unit identifiers to protect the privacy of data contributors [17].

Before receiving access to the eICU database, researchers are required to complete training in human research and data privacy. As recommended by the database managers from the MIT Laboratory for Computational Physiology, the "Data or Specimens Only Research" course, provided by the Collaborative Institutional Training Initiative (CITI Program), was successfully completed. This course contains an overview of ethical and regulatory considerations involved in conducting research using existing data or specimens that are not collected specifically for the research study. It covers topics such as informed consent, data privacy and confidentiality, and the ethical use of existing data or specimens, among other modules. It also provides guidance on how to comply with regulatory requirements and best practices for data management and sharing. Besides the training course, researchers are obliged to agree not to share or attempt to de-identify the data, as well as not release code related to any publication involving the data.

Patient ICU stays assumed a central role in the construction and composition of the

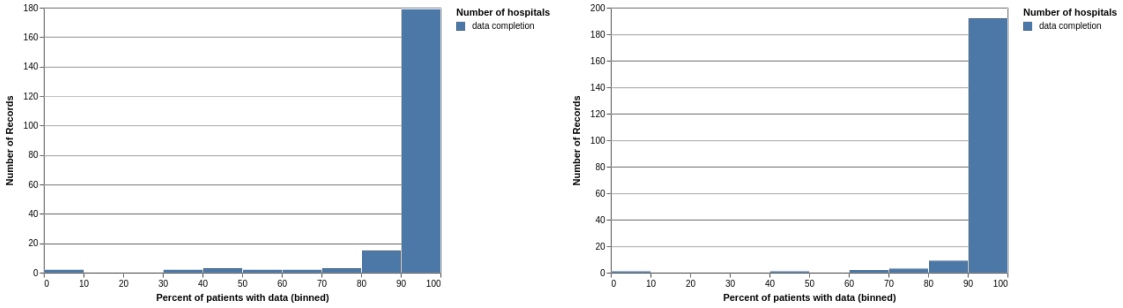
eICU database, providing comprehensive and rich clinical information. The collected data include vital sign measurements, laboratory measurements, medications, care plan information, admission diagnosis, patient history, drug infusions, etc. Across the database, identifiers are used to identify unique concepts and link different tables. These include *hospitalID*, *uniquepid* and *patientHealthSystemStayID* to identify hospitals, patients and hospital stays respectively, though the primary identifier for most tables is *patientUnitStayID*, used to uniquely identify ICU stays. Throughout the database, time is represented through *time stamps* that are measured as offsets from the time of ICU admission.

For the task of early ICU mortality risk prediction with MTS data, the most relevant information consist of vital sign measurements, lab test results, ICU admission and discharge time stamps, ICU discharge patient status and the necessary identifiers. We briefly present the eICU database tables containing this information, which we utilize in our experiments.

- ***patient* table**: It is a core part of the eICU database, as it contains patient demographics and important admission and discharge information, both for hospital and ICU stays, with *patientunitstayid* as the primary identifier. Related to our task, columns *unitdischargestatus* and *unitdischargeoffset* inform us about the ICU stay patient outcome (*Alive* or *Expired*) and the ICU discharge time, as the number of minutes from ICU admission, respectively. Then, the ICU stay is linked via *uniquepid* to a unique patient and via *hospitalid* to a specific hospital. The aforementioned columns are the ones that correspond to the problem studied in this thesis.

- ***vitalperiodic* table**: "Periodic" data refers to data which is consistently obtained from bedside vital signs monitors and loaded into the eCareManager system. The vital signs in the *vitalperiodic* table are not validated by care staff. In order to avoid most cases of spurious observations, data are typically interfaced as 1 minute averages and archived into the table as 5 minute median values. Even though numerous vital signs are observed, we are concerned with heart rate (*heartrate*), respiratory rate (*respiration*), temperature (*temperature*), peripheral oxygen saturation (*saO2*) and systolic, diastolic and mean blood pressure (*systemicsystolic*, *systemicdiastolic* and *systemicmean* respectively). They are linked to a ICU stay via the *patientunitstayid*, while their time stamp is available via the *observationoffset* column.

- ***lab* table**: This table contains laboratory test results that have been mapped to a standard set of measurements. Even though some rarely conducted lab tests are not interfaced into the system, their absence does not necessarily mean they were not conducted. On the other hand, if the result of an interfaced lab test is missing, then this lab test was not conducted. The *labname* and *labresult* columns indicate the laboratory test and the obtained value respectively. In this thesis, we utilize a group of common laboratory tests, namely albumin, band neutrophi, bicarbonate, bilirubin, blood urea nitrogen, chloride, creatinine, glucose, hematocrit, hemoglobin, lactate, partial thromboplastin time, platelet count (platelet), potassium, sodium and



(a) *Vital sign measurements* **(b)** *Lab test results*

Figure 7.1. *Data availability in hospitals [1]*

white blood cells. Similarly to the vitalperiodic table, the columns *patientunitstayid* and *labresultoffset* are employed to link a lab result with an ICU stay and determine its time stamp, in relation to the ICU admission.

As shown in Figure 7.1, the vast majority of the hospitals interfaced vital signs and laboratory tests into the eCareManager system for most stays in their intensive care units. High data coverage is extremely important for successfully training machine learning models on these features.

7.2 Cohort Selection

Let us reiterate the binary classification problem investigated in this thesis. Early ICU mortality risk prediction is defined as the estimation of a patient's risk of dying within a specific timeframe after ICU admission. For this study, we focus on estimating the mortality risk during the second and third day of an ICU stay (that is 24h-72h after ICU admission). The patients who died during their stay in the ICU constitute the positive label group (output=1), while the patients who got discharged from the ICU comprise the negative label group (output=0). Our observation window consists of observations of 7 vital signs and 16 laboratory values, in the form of multivariate time series, obtained during the first 24 hours of an ICU stay. The data within this specific timeframe were extracted and utilized for the purposes of training and evaluating the predictive models.

Regarding the eICU database, an exact time of death is not recorded for patients who died during their ICU stay. As a result, we assume the discharge time stamp corresponds to the end of their ICU stay, similarly to the patients who survived. Moreover, a patient's first vital signs measurements may occur a considerable amount of time after their ICU admission, signifying an ICU admission may be interfaced into the eCareManager system before the patient is set up with clinical monitoring devices. Consequently, we consider the first vital signs observation, after the official ICU admission (that means with an observation offset larger or equal to 0), as the beginning of the ICU stay and take into account only vital signs and laboratory tests that happened after that.

In order to extract a subset of ICU stays for our experiments, we follow the approaches described in Pattalung et al. [104], Mondrejevski et al. [37] and Randl et al. [15] with

some adjustments. The cohort selection criteria applied to the eICU database are the following:

1. Filter for ICU stays with a recorded patient outcome (*Alive* for survival and *Expired* for death).
2. Filter for ICU stays, where the patient's vital signs were observed for at least 24 hours after the beginning of the ICU stay. This condition guarantees that the patient was alive and periodically monitored up until the time of prediction, which is at the 24-hour mark.
3. Filter for ICU stays that lasted at least 24 hours, but not more than 72 hours. This ensures that each patient of the final cohort either survived or died during the second and third day of their ICU stay, and, therefore, is valid for this task.
4. Filter for the first ICU stay of each patient. If a patient is linked with multiple ICU stays at the same hospital, then the first stay can be inferred from the hospital admission offset values for each ICU admission and the hospital admission time. However, the eICU database does not contain specific dates for ICU and hospital stays, making it impossible to determine the first ICU stay of patient among a number of ICU stays across different hospitals. Consequently, we exclude these patients from our study.
5. Filter for ICU stays where at least one laboratory test result is available and interfaced into the eCareManager system.

After applying the above selection criteria, our final study cohort consists of 55,147 ICU stays, each linked with a unique patient. Table 7.1 summarizes the patient demographics for our study cohort. It should be mentioned that the age of patients over 89 years old is not explicitly stated for privacy issues, so, in order to calculate the mean and standard deviation for the age of the cohort, they are assumed to be 90 years old. It is evident that the positive label class (patients who died during their ICU stay) is underrepresented, constituting 3.8% of the total ICU stays of the study cohort. This highlights that our final dataset is *highly imbalanced*, in favor of the negative label class (patients who survived their ICU stay). Consequently, we should take it into account when designing our experiments and evaluation strategy.

7.3 Data Preparation

After selecting the study cohort, the next step involves preparing the data for machine learning training. This data preparation process is crucial to ensure that the data is in a suitable format for training ML models. It typically includes steps such as pruning outliers, re-sampling time series variables, handling missing values and encoding categorical variables. By performing these preparatory tasks, we aim to enhance the quality and reliability of the data, facilitating effective training and evaluation of machine learning algorithms.

Demographics	Total	Survival	Death
Number of Patients	55,147	53,052	2,095
Age (years)	63.72 (17.13)	63.53 (17.16)	68.48 (15.53)
ICU stay (hours)	44 (14.4)	43.96 (14.41)	45.21 (14.04)
Gender			
Male	29,845	28,708	1,137
Female	25,288	24,331	957
Unknown	14	13	1
Ethnicity			
Caucasian	42,548	40,891	1,657
African American	5,837	5,642	195
Hispanic / Latino	2,116	2,054	62
Asian	1,012	966	46
Other / Unknown	3,634	3,499	135

Age and ICU stay values are mean (standard deviation)

Table 7.1. *Patient demographics for our study cohort*

For our work, we are only concerned with the vital signs and laboratory tests that are mentioned in the presentation of the *vitalperiodic* and *lab* tables respectively. First, we remove outliers from these features, which are most likely caused by malfunctions of the monitoring devices. In order to accomplish this, we do not take into account observations that deviate from a valid range of values for each feature. These valid ranges are obtained from the eICU Collaborative Research Database Code Repository [1], as provided by the database managers. By the initial removal of these outliers, we exclude faulty measurements that could tamper with our results.

The following preparatory procedure is to re-sample the variables. The reasons for this are three-fold. First, by re-sampling the time series data in less frequent intervals, the size of the data reduces considerably, facilitating the ML training and evaluation in terms of both time and computational requirements. Then, this process further reduces the effect of spurious readings, by applying a suitable aggregation function on a group of successive observations. Moreover, the re-sampling time intervals are consistent with related work [104, 37, 15], allowing a more direct comparison of different methodologies and results.

Vital signs measurements were typically taken 2.5-12 times per hour, while laboratory test measurements were typically taken 1-2 times per eight hours. Initially, we align the time offsets of the vital signs and lab tests to the beginning of the ICU stay, as we defined it, in contrast to the record of ICU admission. Next, we aggregate each vital signs variable into a 1-hour time interval from the beginning of the ICU stay, while each lab test variable was aggregated into an 8-hour time interval from the beginning of the ICU stay. For every time interval, we calculate a single value for each variable by using the *mean* function on the variable values of the time interval. Up to this point, the re-sampling methodology and selected time intervals correspond to [104, 37, 15]. In addition to this,we use the *min* and *max* functions on three selected vital signs (heart rate, respiratory rate and peripheral oxygen saturation), as they are the most important vital signs features for

mortality prediction in the eICU database, according to Pattalung et al. [104]. In this way, we also address their rich temporal dynamics, which may be partly suppressed due to the re-sampling.

To address phenomena of missing values, we use imputation methods, since most patients are linked to some variables with missing values and their elimination would bias the study. Starting from the beginning of the ICU stay, missing values are filled with forward and, then, backward imputation. However, some variables may not be measured at all, because of the lack of the respective monitoring devices, the severity of the patient's condition or the hospital's decision to not interface this specific variable into the eCareManager system. In this case, the feature's values are replaced with -1, indicating the absence of observations.

Figures 7.2 and 7.3 present the vital signs measurements and the lab test results, during the first 24 hours of an ICU stay, for a patient who got discharged, after the above preparation steps. We observe that temperature was never recorded during this time period, since it constantly has the value -1. Depending on the frequency and the intervals for the original observations, some vital signs are updated every hour, while other measurements change every few hours. Then, many laboratory results have a constant value, indicating that they were conducted only once during this period. Figures 7.4 and 7.5 show the same features for a patient who died during their ICU stay. We observe that many vital signs showcase a different behavior from the previous patient, in terms of the range of values and fluctuations. Lab measurements again do not show many fluctuations, as some tests do not need to be conducted very often.

The last step is encoding the categorical variable of the ICU discharge status to map the ICU stays to the positive and negative classes. As such, we replace the status *Expired* with 1, indicating the positive label class, and the status *Alive* with 0, indicating the negative label class. After following this multi-step procedure, we have formatted our data for ML training and evaluation in our experiments. The overall experimental setup is described in the following chapter.

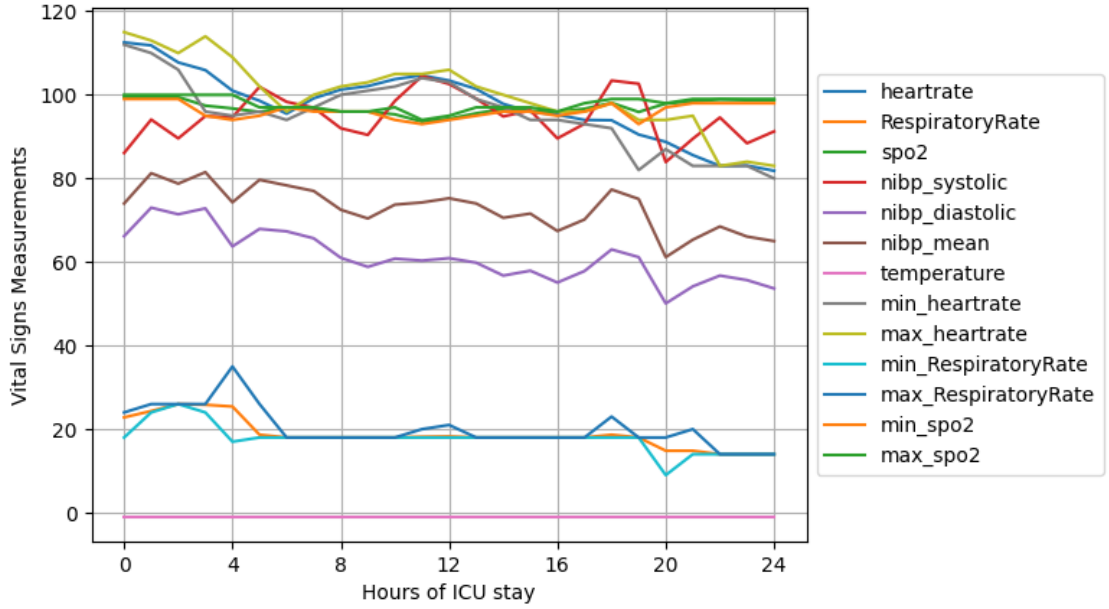


Figure 7.2. *Vital signs measurements during the first 24 hours of an ICU stay, for a patient who survived (output=0).*

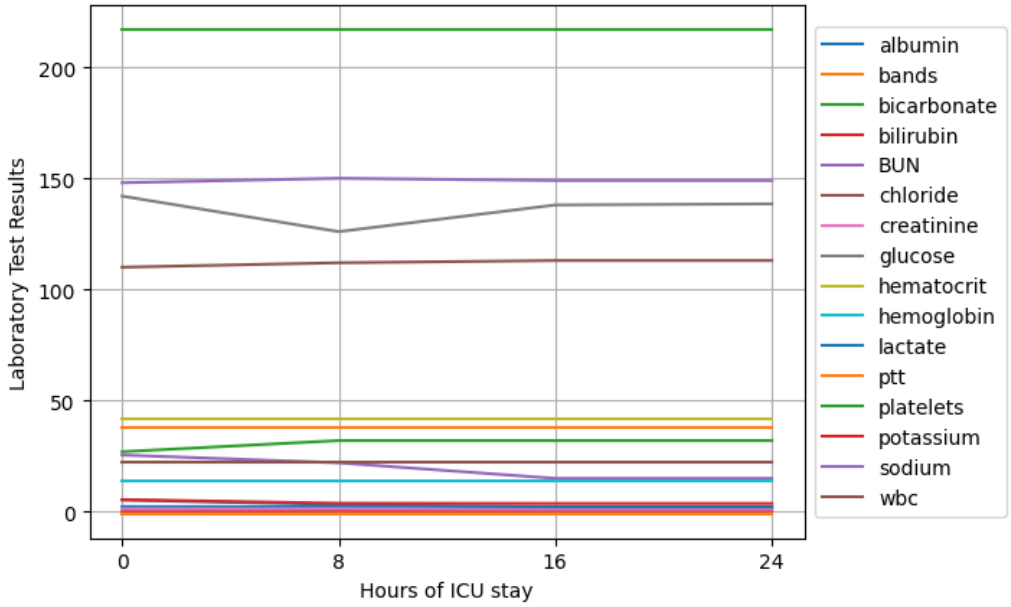


Figure 7.3. *Laboratory test results during the first 24 hours of an ICU stay, for a patient who survived (output=0).*

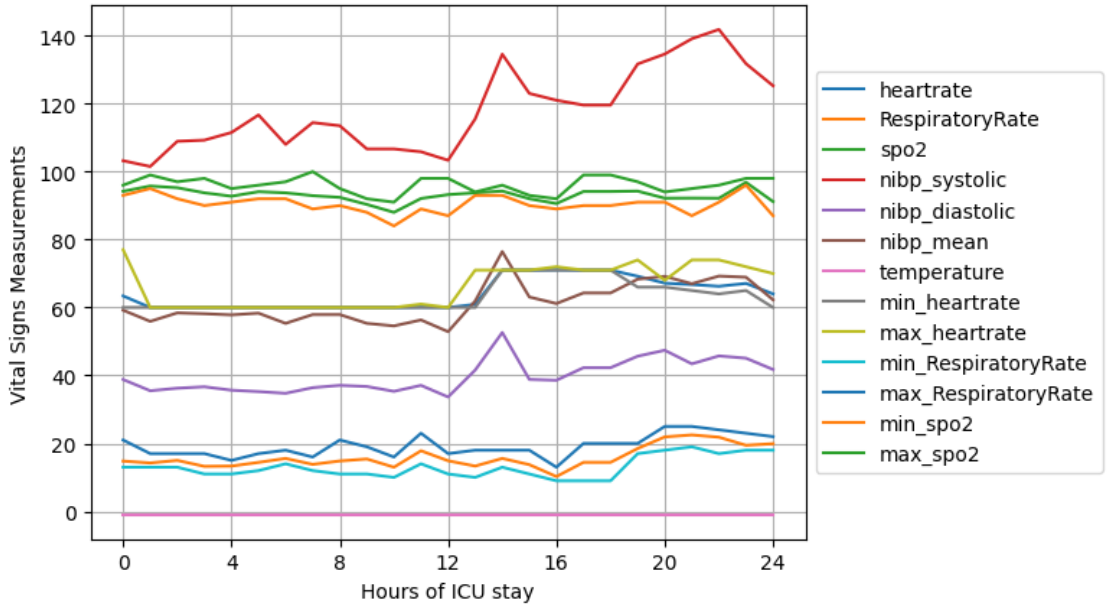


Figure 7.4. *Vital signs measurements during the first 24 hours of an ICU stay, for a patient who died (output=1).*

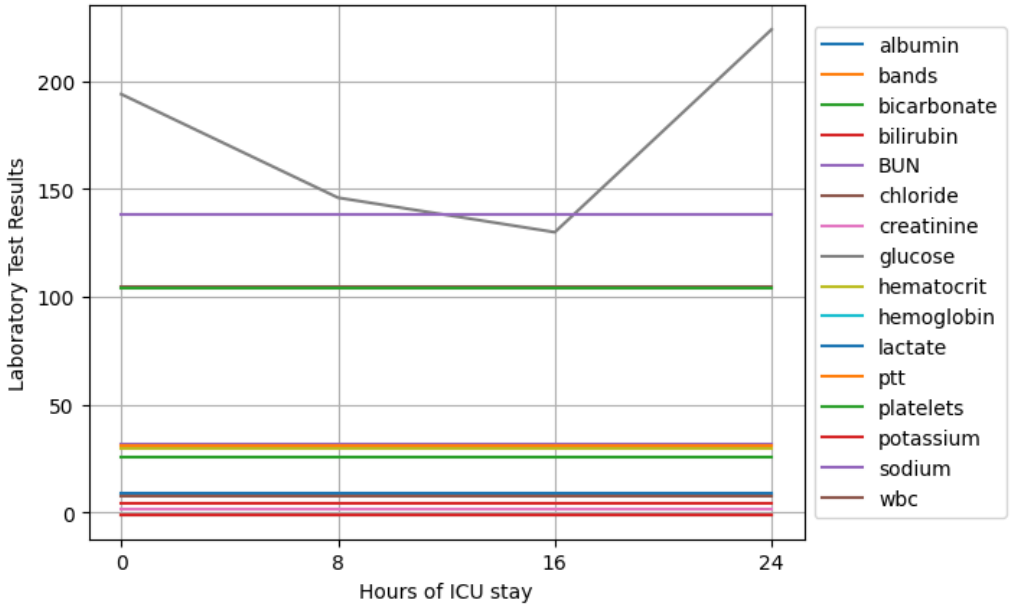


Figure 7.5. *Laboratory test results during the first 24 hours of an ICU stay, for a patient who died (output=1).*

Chapter 8

Experimental Setup

In this chapter, we describe the experimental strategy of our work. First, we introduce the experimental scenarios of this thesis, as well as their accompanying goals. Then, we present the architecture of the constructed neural network models, the design of their training process and the employed metric scores for evaluating their performance on the task of early prediction of ICU mortality risk.

8.1 Experimental Scenarios

This thesis focuses on the application of federated learning in the healthcare domain, addressing the early ICU mortality risk prediction task with the use of MTS data. After careful examination of recent related studies, we designed a series of experimental scenarios that aim to give us an insight into how well federated algorithms address the challenge of statistical heterogeneity in a federated environment. Through our experiments, we analyze the integration of different RNN architectures into the federated learning framework, the performance of current state-of-the-art FL algorithms (FedAvg, FedProx, FedAdam, FedAdagrad, FedYogi, FedAvgM) and the impact of certain FL participants with specific characteristics in the training process. Therefore, we present our experimental FL scenarios and the objectives we aim to achieve with each of them in the following subsections.

8.1.1 Scenario 1: Model Architecture and Generalizability

This first scenario constitutes our basic experimental setup. Due to the high computational and memory complexity of a simulated FL environment, we select 8 hospitals from our study cohort to comprise our FL training participants. This allows us to experiment more with different models and setups, facilitates the analysis of the different factors that affect federated learning and enables a direct comparison of our results with those of Randl et al. [15]. Our selection is based on the diversity of the patient populations and demographics, in order to ensure the statistical heterogeneity of our local training datasets. Table 8.1 summarizes the data distributions of each single training hospital and the training set as a whole. It is evident that the datasets of the participating hospitals differ in terms of size and class distribution, highlighting the non-IID settings of our experiments.

Datasets	Total	Survival	Death
Hospital A	1,018	977	41 (4.0%)
Hospital B	1,041	972	69 (6.6%)
Hospital C	1,788	1,714	74 (4.1%)
Hospital D	773	746	27 (3.5%)
Hospital E	1,129	1,088	41 (3.6%)
Hospital F	1,344	1,244	100 (7.4%)
Hospital G	930	878	52 (5.6%)
Hospital H	1,316	1,248	68 (5.2%)
Training Set	9,339	8,867	472 (5.1%)

Death column also contains (% over total)

Table 8.1. *Data distributions for the participating training hospitals in Scenario 1*

There are two goals linked with this scenario. The first one is to evaluate the extent of the generalizability of the obtained models with each FL algorithm, by comparing them with the respective models of the CML and LML approaches. With regard to this, we use a foreign test set, consisting of patient data from 4 hospitals that do not participate in the training process. Table 8.2 summarizes the data distributions of each single testing hospital and the testing set as a whole. By combining data from 4 different hospitals to assemble the test set, we may gain a better understanding of the ability of each FL algorithm to utilize information from all FL clients to build a robust, effective model.

Datasets	Total	Survival	Death
Hospital I	421	398	23 (5.5%)
Hospital J	477	456	21 (4.4%)
Hospital K	420	397	23 (5.5%)
Hospital L	485	460	25 (5.2%)
Test Set	1,803	1,711	92 (5.1%)

Death column also contains (% over total)

Table 8.2. *Data distributions for the test set in Scenario 1*

The second goal is to evaluate the incorporation of different RNN models (RNN, LSTM, GRU) into the federated learning framework. We do not limit ourselves to comparing their performance on the chosen evaluation metrics, but we also aim to assess the computational overhead of each architecture. In order to achieve this, we take into account the number of trainable parameters that are transmitted over the network and the average FL rounds until convergence. Overall, this scenario aims to assess the ability of each FL algorithm to develop generalizable models and their integration with different RNN architectures.

8.1.2 Scenario 2: 'Extreme Cases' of Participating Hospital

Our second experimental scenario considers the robustness of the different FL methods under the presence of artificial 'extreme' FL participants, in terms of data size and class distribution. Our FL environment consists of the 8 training hospitals from the first scenario and each of two 'extreme' hospitals, which we construct by aggregating ICU stays

from the remaining hospitals. *Hospital X1* includes the most ICU stays, compared to the rest 8 training hospitals, however it is characterized by an extreme class imbalance, with just above 0.5% of their ICU stays in the positive class (mortality). As the FL participant with the most data, hospital X1 influences the aggregated FL model more than any other FL client and our goal is to regulate this effect. On the other hand, *hospital X2* includes data only for 300 ICU stays with 25% of them belonging to the positive class. This positive class distribution is much higher than for any other training hospital (hospital F is the second highest with 7.6%). As this hospital contains much useful information for the under-represented positive class, our aim is to process and utilize its information in the best possible way. Table 8.3 summarizes the data distributions of each 'extreme' hospital and the derived 9-hospital training sets for these experiments.

Datasets	Total	Survival	Death
Hospital X1	1,900	1,890	10 (0.5%)
Training Set + X1	11,239	10,757	482 (4.3%)
Hospital X2	300	225	75 (25.0%)
Training Set + X2	9,639	9,092	547 (5.7%)

Death column also contains (% over total)

Table 8.3. *Data distributions for the datasets in Scenario 2*

For these experiments, we split the 9-hospital datasets into training, validation and test sets with sizes 68%, 12% and 20% of the original datasets respectively. Overall, the goal of this experiment is to evaluate the sensitivity of each FL algorithm in 2 FL environments that require special attention. Either with a highly influential FL client with a negative-biased dataset, or with a limited FL client with valuable information, these FL environments provide serious challenges to the proposed FL methods.

8.1.3 Scenario 3: Importance of Participation

One of the cornerstones of federated learning is the participation of a party in the training process and the construction of a ML model, along with other parties. In the first scenario, we evaluate the performance of the final model on a foreign test set and, in the second scenario, on a test set assembled by data from the FL participants. This third experimental scenario evaluates the performance of the FL models on the test set of specific FL clients. Our FL environment consists once again of the 8 training hospitals from scenario 1, however we focus on two hospitals G and C, the former with fewer data and the latter with the most data within this training set. These two hospitals will serve as our scenario subjects, each of them providing as with a test set for our experiments.

We explore the performance of FL on each of these two hospitals in an isolated sub-scenario. In the first sub-scenario, we use 20% of the hospital G's data as a test set and consider 3 different approaches. The first is the LML approach, with hospital G training its own local model. The second is the privacy-preserving FL approach without the participation of hospital G (utilized with *transfer learning*). Finally, the third is the FL approach with the participation of hospital G. Through this experiment, we want to measure the

impact of hospital G's participation in the FL setup on its own test set, compared to the other two aforementioned approaches. The other sub-scenario concerns hospital C in a similar manner. By experimenting with two hospitals with different characteristics, such as hospital G and C, we aim to draw conclusions with regard to how much a hospital with adequate or insufficient amount of data can be benefited from participating in a FL environment.

8.2 Model Architecture & Optimization

As mentioned earlier throughout this thesis, the input data of this task are vital signs and laboratory results, in the form of multivariate time series, from the first 24 hours of a patient's ICU stay. The temporal nature of the data dictates the use of sequential neural network architectures, which are capable to learn complex temporal dynamics from the input. With regard to this, we employ recurrent neural network architectures (RNNs) as the basic building block of our predictive model.

Our model consists of two parallel input channels, one of them processing the vital signs measurements and the other processing the laboratory test results. The input data have already been pre-processed and prepared for training, as we discussed in the previous chapter. Each of these branches is implemented with 3 recurrent layers of 16 units each, conferring a sense of depthness on the model and allowing the progressive abstraction of the information and its transformation into high-level representations. Each of these two branches is followed by batch normalization, before we concatenate their outputs along the channel dimension. Subsequently, two fully-connected layers (with 8 outputs and 1 output respectively) fuse these feature maps and a sigmoid layer applies a sigmoid function to the single, final output to calculate the risk of mortality in the ICU in the range [0,1]. We explore 3 different variants of RNNs (RNN, LSTM, GRU), leading to different amounts of trainable parameters, shown in Table 8.4. The number of trainable parameters is of great importance in a federated learning setting, as it signalizes the communication overhead of the whole training procedure, in accordance with the total FL training rounds and the number of FL clients.

Model	Trainable Parameters
RNN	4,001
LSTM	13,361
GRU	10,529

Table 8.4. *Trainable parameters for our model with each RNN architecture*

Since we aim to predict whether a patient is going to die or not during their ICU stay, we use the binary cross-entropy loss function and the Adaptive Moment Estimation (Adam) optimization algorithm, which are suitable optimization methods for binary classification tasks.

8.3 Training Process

We evaluate three different approaches, CML, LML and FL, on the three experimental scenarios we described earlier. Depending on the scenario, the testing data could either be a foreign test set, an aggregated test set from the training hospital cohort or an individual test set from a specific hospital. For the foreign test set, we use 5-fold cross-validation on the training-validation splits to eliminate any randomness induced by data partitioning. For the two latter test sets, we use 5-fold cross-validation on the test splits, allocating 20% of the data for testing, and then split the remaining data into an 85% training set and a 15% validation set. As our data are, in fact, an aggregation of several local datasets, the above procedure is applied to every local dataset individually. Moreover, for every approach, the training models are initialized with the same randomized weights to tackle any additive bias from choosing different initial weights as the models' starting point in each experiment.

For all three CML, LML and FL approaches, we initially normalize the training and validation datasets before the training commences. Then, we address the problem of class imbalance by assigning class weights to data instances during training, in order to give equal importance to the gradient updates of both classes. The CML models are trained on a mini-batch size of 256, while the local and FL models are trained on a mini-batch size of 32, respective to the size of local datasets compared to the size of the aggregated global dataset. Training starts with a local initial learning rate of 0.001, which drops by 50% every 5 epochs, in order to avoid undesirable divergent behavior with the loss function. Furthermore, we implement an early stopping mechanism, monitoring the F1-Score on the validation set, which presents a more defined maximum during validation than the other metrics, with a patience period of 30 epochs and a maximum training period of 100 epochs. When the early stopping mechanism is triggered, we restore the weights of the best model during training, in terms of F1-Score on the validation set.

Since federated learning involves more hyperparameters, it requires additional setup steps. All FL clients are trained for 1 epoch per FL round, in order to maintain a high training speed. We assume total participation of the federated network, which corresponds to the real-life scenario, since hospitals are typically considered powerful and reliable computational nodes. The weight aggregation of the standard FedAvg algorithm is based on the relative sizes of the FL participants. The FedProx algorithm involves a proximal term that regulates the local model updates, which is set to 0.1 after calibrations within our experiments. Regarding the server-side optimization FL algorithms, the decay rates β_1 and β_2 are set to 0.9 and 0.99 respectively, while the server-side learning rate and τ are adjusted based on the experimental setting. Similarly, the learning rate of FedAvgM is adjusted to each experiment, while the momentum factor of the optimization step is set to 0.9. The overall FL training design and the rest of the training parameters are the same as the ones presented above for the CML and LML approaches.

8.4 Evaluation Metrics

The performance of each model is evaluated on a test set dictated by the chosen experimental scenario. Either it is a foreign test set (scenario 1), a test set consisting of data from all participating FL clients (scenario 2), or a test set with data from a specific hospital (scenario 3). Moving forward, we consider a true positive (TP) as the result of a correct mortality prediction, a true negative (TN) as the result of a correct discharge prediction, a false positive (FP) as the result of a wrong mortality prediction and a false negative (FN) as the result of a wrong discharge prediction.

Typically, for ML tasks, accuracy is the most commonly used evaluation metric, defined as the proportion of correct classifications over the total predictions. However, on imbalanced problems, accuracy can be a misleading criterion for the performance of a model, since it could achieve high accuracy by simply predicting only the over-represented class, thus being unable to capture the patterns of the under-represented class and generalize. As a result, we need to utilize other metrics to effectively assess the model's performance and better understand its characteristics. In order to do so, we first define some basic evaluation metrics:

$$Recall = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP}$$

$$NegativePredictiveValue(NPV) = \frac{TN}{TN + FN}$$

Recall indicates how many of the total positive class instances were correctly identified, while *specificity* is its symmetrical counterpart regarding the negative class instances. Then, *precision* measures the ability of the model to correctly classify the positive class, whereas the analogous *negative predictive rate* is not often employed for model evaluation.

By plotting recall and specificity at different classification threshold values, one can generate the *receiver operating characteristic (ROC)* curve. The area under the ROC (AUC) is a metric that describes the overall classification performance of a predictive model and it constitutes the main focus of many related studies [104, 14], with respect to model evaluation. Respectively, by plotting precision against recall in a similar manner, one can generate the *Precision-Recall curve (PRC)* and utilize the area under the PRC (AUPRC) as a performance metric. PRC visualizes the precision-recall tradeoff, a frequent challenge for predictive tasks, allowing researchers to find the optimal classification threshold in accordance with the goals of their experiments. Another metric that is expressed as a relationship between precision and recall is *F1-Score*, which is the harmonic mean

between the two. AUPRC and F1-Score are most informative about a model's performance, when identifying the positive class correctly is more important than predicting the negative class.

In our work, we evaluate the performance of our models on these three aforementioned metrics (AUROC, AUPRC and F1-Score). As we address a binary classification task with a highly imbalanced dataset, we tend to focus more on AUPRC and F1-Score. The results of the AUROC metric are often unreliable, when the minority class is heavily under-represented, while the AUPRC is more suitable for such problems and provides more useful information for the behavior of a model [38]. Besides these metric scores, we also monitor the training rounds that each FL algorithm requires until achieving the best F1-Score on the validation set, as the communication overhead is an important aspect of the FL setting.

Chapter 9

Results & Discussion

This chapter presents the experimental results and discussion pertaining to the evaluation of the proposed methodologies for early prediction of ICU mortality risk using federated learning with multivariate time series data. Our main objectives are to examine the performance of different RNN models integrated within the federated learning framework, conduct a comparative analysis of various state-of-the-art federated learning algorithms and explore the impact of participation in a FL setting for individual clients.

The first experimental scenario focuses on evaluating the FL models on a foreign test set, thus exploring their generalizability and the degree of incorporation of each RNN model within the FL environment. Then, the second experiment investigates the sensitivity and robustness of FL algorithms in federated learning environments with varying degrees of statistical heterogeneity, by including a ninth artificially constructed hospital with 'extreme' data distribution into the FL training cohort. The third scenario focuses on two specific hospitals and measures the effect of their participation, in training a FL model, on their local test sets, thus indicating the motivation for potential hospitals to engage in a federated learning environment.

Through the discussion on the results, we provide a comprehensive analysis of the performance and limitations of the proposed approaches. We aim to gain a deeper understanding of the attributes of a federated learning environment and the potential of federated learning for early prediction of mortality risk, which may expand to other problems both within the healthcare domain and in other industries and fields of interest

9.1 Results for Experimental Scenario 1

9.1.1 RNN architecture

Table 9.1 contains the metric scores for the CML, LML and FL approaches with the RNN model. The CML model, which treats the local datasets as a single, centralized dataset, achieved an AUROC of 0.836, an AUPRC of 0.400 and a F1-Score of 0.473. These scores are considerably better than the ones obtained with the localized models, which reached an AUROC of 0.764, an AUPRC of 0.239 and a F1-Score of 0.252 respectively. Overall, all FL models outperformed the LML models by a significant margin. In terms of AUROC, the FedProx algorithm, with a score of 0.838, outperformed all the other FL algorithms, as well as the CML model, while FedAdam had a similar performance, with a score of 0.837. Then, FedAdam achieved the best AUPRC among the FL models, with a score of 0.364. However, the F1-Score achieved with FedAdam (0.303) was the worst among the FL models, whereas the standard FedAvg, with a F1-Score of 0.335, was the best performing FL model. Besides F1-Score, the FedAvg obtained good AUROC and AUPRC scores as well, narrowly behind FedProx and FedAdam that led in these metrics. The remaining server-side optimization FL algorithms, namely FedAdagrad, FedYogi and FedAvgM, obtained the lowest AUROC and AUPRC scores between the FL models, while their F1-Scores were comparable to FedProx. The ROC and Precision-Recall curves obtained with the CML, LML and FL approaches and RNN-based models are visualized in Figures 9.1 and 9.2 respectively.

Regarding the FL training rounds, we observe that the best F1-Score on the validation set was achieved, on average, after 10 rounds for most FL algorithms. FedProx required the most training until convergence, achieving it after 12.5 FL rounds, on average, while FedAvg required 11.2 FL rounds respectively. Among the server-side optimization algorithms, FedAdam reported the best F1-Score on the validation set after 9.8 FL rounds, on average, while the remaining FL algorithms trained for approximately 9 FL rounds. This could possibly partly explain their lower AUROC and AUPRC scores in comparison to the other FL algorithms, as they may have obtained them slightly earlier than the optimal training time. Overall, the performance of the FL algorithms was significantly better than the one achieved with the LML approach, while they achieved comparable AUROC and AUPRC with the CML approach.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.836 ± 0.013	0.400 ± 0.009	0.473 ± 0.029	-
LML	0.764 ± 0.009	0.239 ± 0.009	0.252 ± 0.013	-
FedAvg	0.832 ± 0.037	0.348 ± 0.052	**0.335 ± 0.047**	11.2 ± 4.3
FedProx	**0.838 ± 0.021**	0.360 ± 0.039	0.321 ± 0.035	12.5 ± 3.4
FedAdam	0.837 ± 0.022	**0.364 ± 0.037**	0.303 ± 0.007	9.8 ± 1.2
FedAdagrad	0.815 ± 0.016	0.318 ± 0.016	0.327 ± 0.024	9.3 ± 2.3
FedYogi	0.825 ± 0.021	0.329 ± 0.029	0.325 ± 0.041	9.1 ± 1.1
FedAvgM	0.818 ± 0.023	0.322 ± 0.035	0.314 ± 0.035	8.7 ± 2.2

Table 9.1. *Test Performance with RNN architecture*

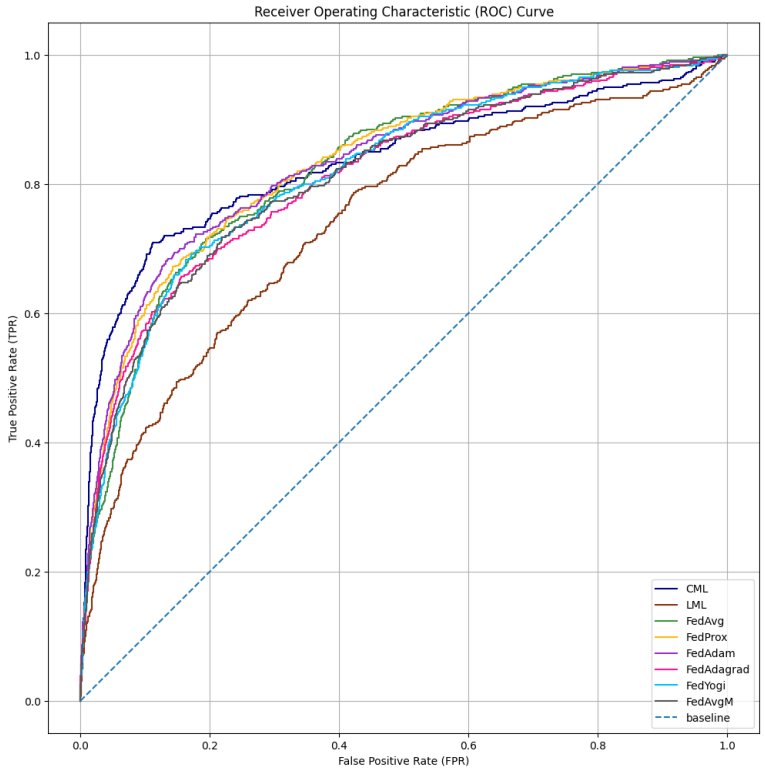


Figure 9.1. *Comparison of ROC curves obtained with CML, LML and FL approaches with the RNN architecture*

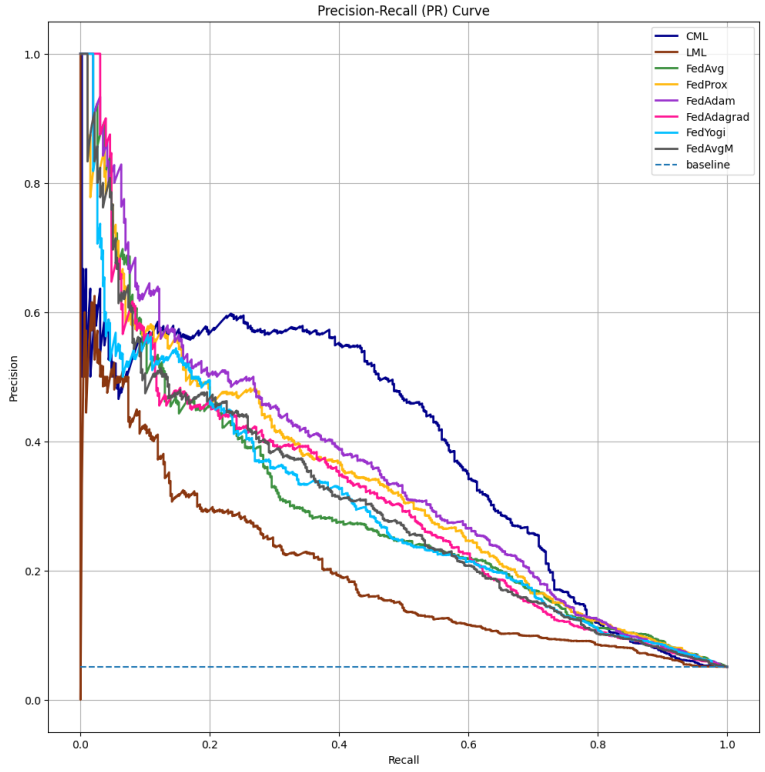


Figure 9.2. *Comparison of Precision-Recall curves obtained with CML, LML and FL approaches with the RNN architecture*

9.1.2 LSTM architecture

Table 9.2 contains the metric scores for the CML, LML and FL approaches with the LSTM model. The ideal CML model achieved an AUROC of 0.894, an AUPRC of 0.499 and a F1-Score of 0.489. The LML approach, which treats each hospital as an isolated data silo and trains a local model for each hospital, obtained an AUROC of 0.803, an AUPRC of 0.326 and a F1-Score of 0.384. Among the FL algorithms, FedAvg provided the best AUROC score, but the worst AUPRC score, 0.899 and 0.446 respectively. It is important to note that all FL models, except FedYogi, achieved better AUROC scores than the CML model, with FedProx, FedAdagrad and FedAvgM roughly equalizing the score of FedAvg with an AUROC of 0.897. The best AUPRC score among the FL methods was obtained with FedYogi (0.476), approximating the AURPC of the CML model, while the FedProx, FedAdam and FedAvgM models reached AUPRC scores ranging from 0.453 to 0.456, considerably lower than FedYogi. Regarding F1-Score, FedProx performed the best with a score of 0.438, whereas FedYogi and FedAdagrad followed up with scores 0.424 and 0.422 respectively. Once again, all FL models outperformed the LML models in every metric category, showing a better predictive behavior on the foreign test set. The ROC and Precision-Recall curves obtained with the CML, LML and FL approaches and LTSM-based models are visualized in Figures 9.3 and 9.4 respectively.

We observe that the LSTM models required further FL training rounds before triggering the early stopping mechanism, which is probably caused by the increased number of trainable parameters (the LSTM model involves over three times more parameters than the RNN model). Again, FedProx required the most FL training rounds until convergence, 18.9 on average, followed by FedAvgM with an average of 17.2 FL rounds and FedAvg with an average of 16.8 FL rounds. The remaining federated optimization algorithms achieved convergence in less FL rounds, ranging from 14.7 to 16.5 on average. The models constructed following all 3 learning approaches were benefited from the LSTM architecture, achieving better scores for every metric. However, the FL models showed the most significant improvement, in comparison with the CML approach, as their reported AUPRC and F1-Scores were closer to the ideal scores of the CML model.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.894 ± 0.007	0.499 ± 0.020	0.489 ± 0.023	-
LML	0.803 ± 0.034	0.326 ± 0.038	0.384 ± 0.020	-
FedAvg	**0.899 ± 0.006**	0.446 ± 0.037	0.417 ± 0.035	16.8 ± 3.9
FedProx	0.897 ± 0.010	0.456 ± 0.032	**0.438 ± 0.014**	18.9 ± 3.0
FedAdam	0.895 ± 0.009	0.454 ± 0.029	0.409 ± 0.028	16.5 ± 1.8
FedAdagrad	0.897 ± 0.010	0.449 ± 0.035	0.422 ± 0.036	15.4 ± 1.7
FedYogi	0.893 ± 0.010	**0.476 ± 0.022**	0.424 ± 0.040	14.7 ± 0.8
FedAvgM	0.897 ± 0.011	0.453 ± 0.038	0.418 ± 0.031	17.2 ± 3.8

Table 9.2. *Test Performance with LSTM architecture*

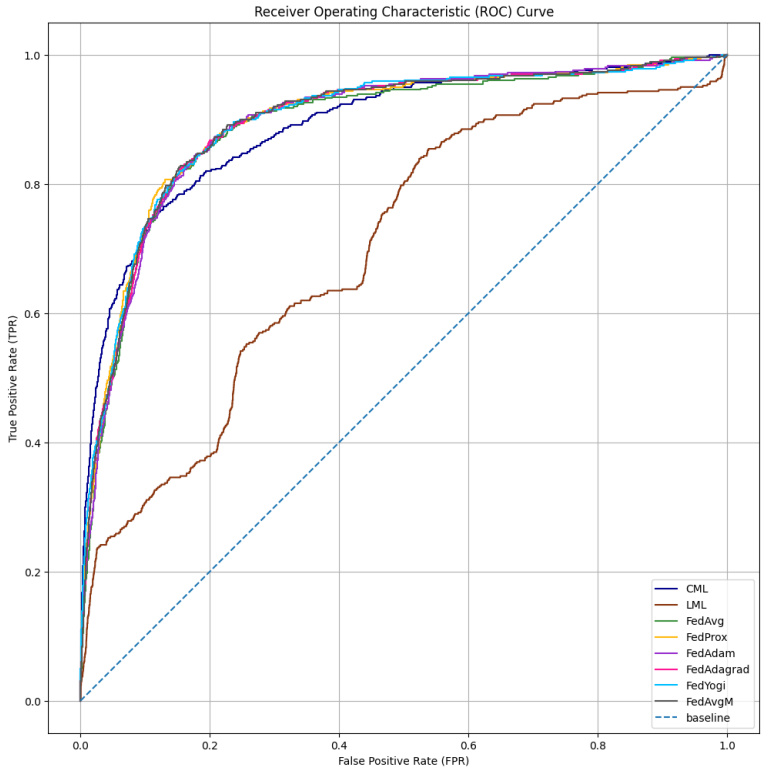


Figure 9.3. *Comparison of ROC curves obtained with CML, LML and FL approaches with the LSTM architecture*

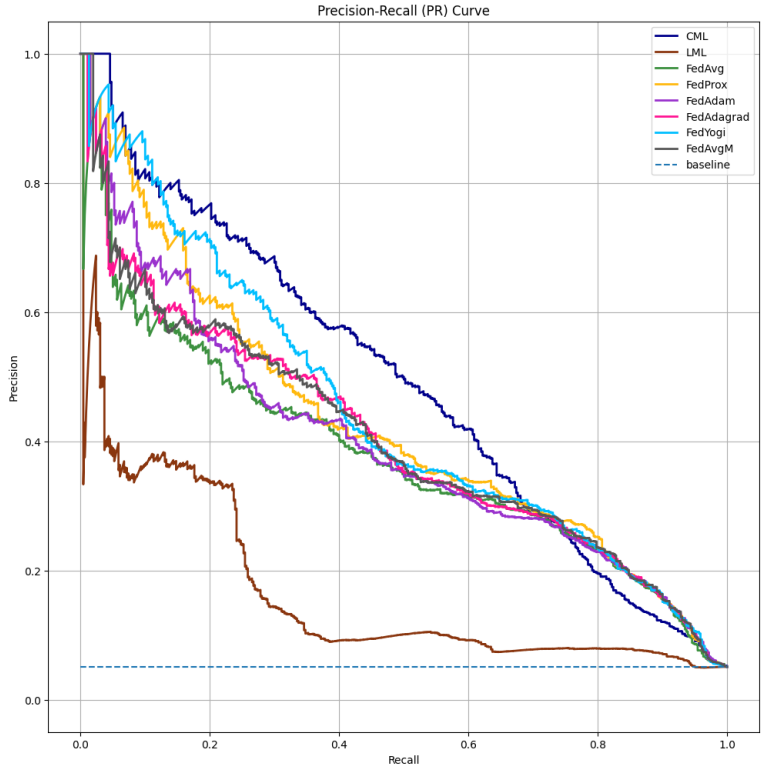


Figure 9.4. *Comparison of Precision-Recall curves obtained with CML, LML and FL approaches with the LSTM architecture*

9.1.3 GRU architecture

Table 9.3 contains the metric scores for the CML, LML and FL approaches with the GRU model. The model constructed with the CML approach obtained an AUROC of 0.895, an AUPRC of 0.539 and a F1-Score of 0.541. On the other hand, the LML approach with the localized hospital models achieved an AUROC of 0.807, an AUPRC of 0.360 and a F1-Score of 0.541, deviating significantly from the metric scores of CML. With regard to AUROC, the FedAdagrad optimization algorithm reported the highest score, specifically 0.892, while FedAdam, FedProx and FedAvg roughly achieved an equivalent score. In terms of AUPRC, the model trained with the FedProx algorithm reached a score of 0.507, the FedAdam model achieved a score of 0.505 and the remaining FL algorithms had a similar performance ranging from 0.499 to 0.502. The FedAdagrad algorithm also led to the best F1-Score among the FL methods, reporting a score of 0.512. For this metric, the other FL algorithms scored considerably lower, with FedAvg scoring 0.489 and the remaining methods ranging from 0.466 to 0.480. Overall, the FL GRU-based models performed significantly better than the LML models, while they achieved better AUPRC and F1-Scores than the respective LSTM-based models. The ROC and Precision-Recall curves obtained with the CML, LML and FL approaches and GRU-based models are visualized in Figures 9.5 and 9.6 respectively.

Considering the number of FL training rounds, we observe that the FL algorithms required on average less FL rounds than the LSTM-based models until convergence, but considerably higher than those of the RNN-based models. In these experiments, FedAdam achieved the best F1-Score on the validation set after 16.1 FL rounds on average, while FedAdagrad and FedAvg required 15.5 and 15.2 FL rounds on average respectively. Then, FedYogi, FedAvgM and FedProx reported their best FL model approximately after 14 FL training rounds. Taking everything into account, the GRU-based models achieved a comparable AUROC to the LSTM-based models, however they reported an improvement in terms of AUPRC and F1-Score.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.895 ± 0.002	0.539 ± 0.004	0.541 ± 0.020	-
LML	0.807 ± 0.032	0.360 ± 0.021	0.413 ± 0.024	-
FedAvg	0.890 ± 0.007	0.499 ± 0.020	0.489 ± 0.023	15.2 ± 2.3
FedProx	0.891 ± 0.006	**0.507 ± 0.016**	0.472 ± 0.042	14.2 ± 2.5
FedAdam	0.891 ± 0.005	0.505 ± 0.014	0.480 ± 0.046	16.1 ± 2.1
FedAdagrad	**0.892 ± 0.006**	0.502 ± 0.018	**0.512 ± 0.032**	15.5 ± 1.7
FedYogi	0.887 ± 0.007	0.500 ± 0.021	0.475 ± 0.030	13.7 ± 0.7
FedAvgM	0.889 ± 0.008	0.502 ± 0.020	0.466 ± 0.039	13.9 ± 0.9

Table 9.3. *Test Performance with GRU architecture*

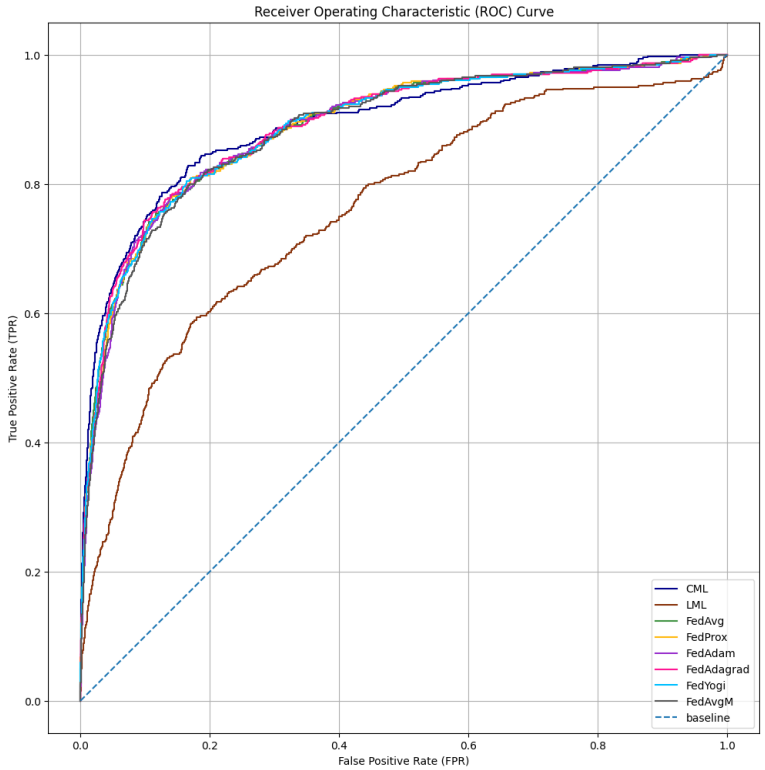


Figure 9.5. *Comparison of ROC curves obtained with CML, LML and FL approaches with the GRU architecture*

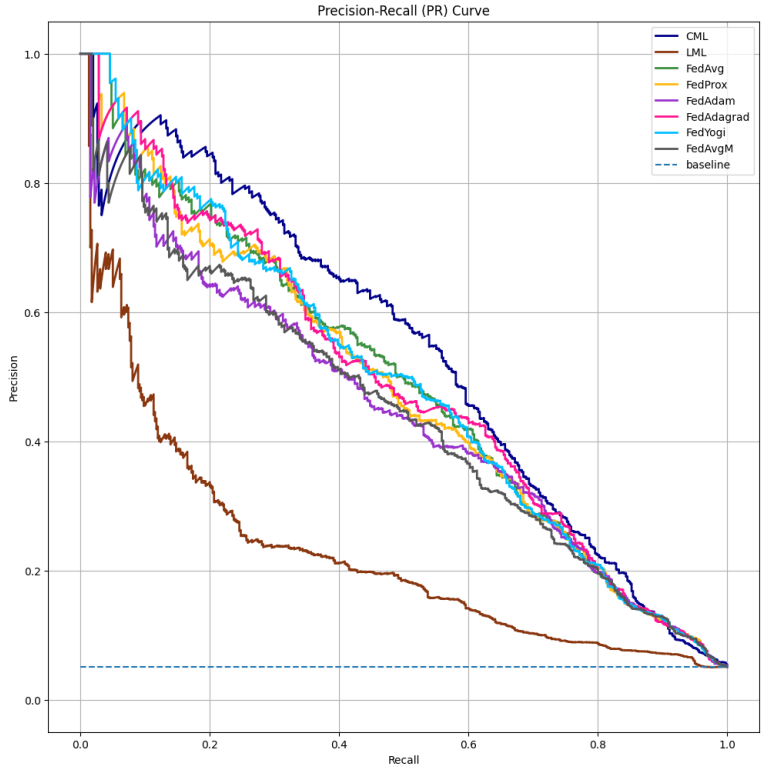


Figure 9.6. *Comparison of Precision-Recall curves obtained with CML, LML and FL approaches with the GRU architecture*

9.2 Results for Experimental Scenario 2

9.2.1 Extreme Hospital X1

Table 9.4 contains the metric scores for the CML, LML and FL approaches with the participation of *hospital X1* in the FL training process. The CML model, our point of reference, achieved an AUROC of 0.870, an AUPRC of 0.461 and a F1-Score of 0.498. The local models were heavily affected by the increased statistical heterogeneity, leading to an AUROC of 0.689, an AUPRC of 0.185 and a F1-Score of 0.289. In terms of AUROC, all FL models performed slightly worse than the CML model, with FedProx achieving the best AUROC among them (0.854). FedAvg follows up with an AUROC score of 0.852, while the remaining FL algorithms reported AUROC scores in the range of 0.0848-0.851. Regarding AUPRC, the FL algorithms were considerably affected by the presence of hospital X1 among the participants, with FedAvg performing the best with an AUPRC of 0.405. The second best performing FL algorithm was FedProx with an AUPRC of 0.402, while the server-side optimization FL algorithms reached an AUPRC ranging from 0.390 to 0.392, besides FedYogi with an AUPRC of 0.399. Considering the F1-Scores, two of the server-side optimization FL algorithms, FedYogi and FedAdagrad, stand out, recording a F1-Score of 0.433 and 0.426 respectively, while FedProx and FedAvg achieved a F1-Score of 0.413 and 0.407. Overall, each approach was affected by this new experimental environment, with the performance of the FL methods slightly deviating from the scores of the CML approach, compared to the previous experiment with the GRU-based models. The ROC and Precision-Recall curves obtained with the CML, LML and FL approaches and the participating hospital X1 are visualized in Figures 9.7 and 9.8 respectively.

Observing the best FL round of each FL algorithm, the first thing to notice is that the FL methods trained longer before triggering the ES mechanism, compared to the previous experiment. FedAdagrad required 17.2 FL training rounds on average, while FedProx and FedYogi needed more than 16 FL rounds before reporting their best F1-Score on the validation set. The other FL methods converged approximately after 15.5 FL rounds on average. Taking everything into account, the participating hospital X1 altered the characteristics of the global dataset and the FL environment, affecting every ML approach. The FL methods still outperformed the LML approach, even though they deviated more from the ideal CML performance.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.870 ± 0.004	0.461 ± 0.046	0.498 ± 0.042	-
LML	0.689 ± 0.030	0.185 ± 0.028	0.289 ± 0.046	-
FedAvg	0.852 ± 0.008	**0.405 ± 0.055**	0.407 ± 0.058	15.5 ± 2.3
FedProx	**0.854 ± 0.008**	0.402 ± 0.044	0.413 ± 0.47	16.5 ± 1.2
FedAdam	0.851 ± 0.012	0.392 ± 0.056	0.400 ± 0.072	15.6 ± 2.0
FedAdagrad	0.848 ± 0.007	0.390 ± 0.049	0.426 ± 0.060	17.2 ± 2.4
FedYogi	0.850 ± 0.008	0.399 ± 0.051	**0.433 ± 0.050**	16.1 ± 1.4
FedAvgM	0.851 ± 0.006	0.392 ± 0.042	0.395 ± 0.074	15.7 ± 2.5

Table 9.4. *Test Performance with 'extreme' hospital X1*

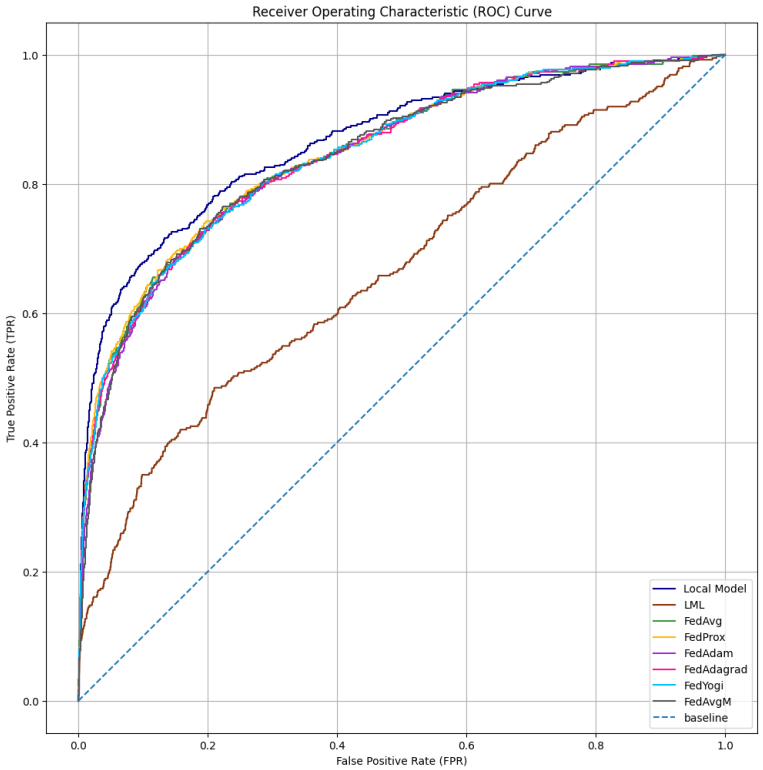


Figure 9.7. *Comparison of ROC curves obtained with CML, LML and FL approaches with the 'extreme' hospital X1*

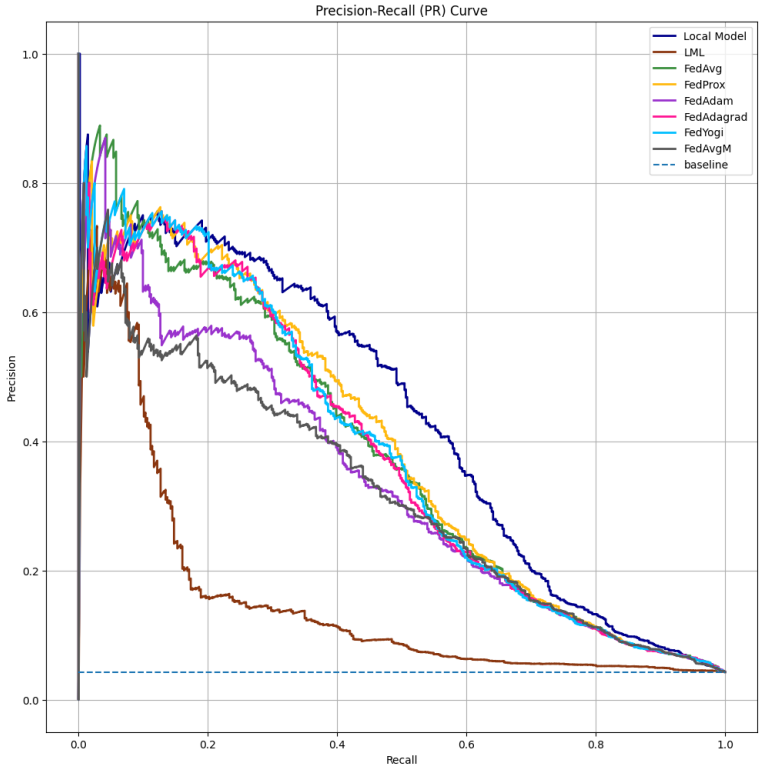


Figure 9.8. *Comparison of Precision-Recall curves obtained with CML, LML and FL approaches with 'extreme' hospital X1*

9.2.2 Extreme Hospital X2

Table 9.5 contains the metric scores for the CML, LML and FL approaches with the participation of *hospital X2* in the FL training process. The model trained with the CML approach achieved an AUROC of 0.871, an AUPRC of 0.537 and a F1-Score of 0.526. The local models performed again significantly worse with an AUROC of 0.728, an AUPRC of 0.288 and a F1-Score of 0.390. The FL models maintained a performance relatively closer to that of the centralized ML method, both in terms of AUROC and AUPRC. The best performing FL algorithm in these two metric scores was FedAvgM, with 0.860 and 0.503 respectively. Regarding AUROC, the remaining FL algorithms showed a similar performance ranging from 0.854 (FedYogi) to 0.859 (FedAvg and FedProx). With respect to AUPRC, FedYogi reported the second best score (0.501), while the remaining FL algorithms obtained scores ranging from 0.492 to 0.495, besides the standard FedAvg method (0.481). Moreover, FedYogi achieved the best F1-Score (0.477), followed by the rest of the server-side optimization FL methods, with FedAvgM and FedAdagrad reaching a F1-Score of 0.463. FedProx considerably underperformed in comparison to the other FL algorithms, achieving a F1-Score of 0.405, a slight improvement upon the F1-Score of the LML approach. On average, the FL methods effectively handled the additive statistic heterogeneity of hospital X2, maintaining their performance with reference to the CML approach. The ROC and Precision-Recall curves obtained with the CML, LML and FL approaches and the participating hospital X2 are visualized in Figures 9.9 and 9.10 respectively.

Considering the number of FL training rounds for each FL algorithm, we observe that all FL algorithms required more FL training on average than in the previous experiments, before reaching the best F1-Score on the validation set and triggering the early stopping mechanism. The FedAvgM method required the most FL rounds on average (20.2), while FedAdam needed the least for convergence (19.1). Overall, we observe that CML obtained comparable metric scores to our previous basic experiment with the GRU-based models, while the FL algorithms were slightly affected in a negative manner.

Method	AUROC	AUPRC	F1-Score	Best FL Round
CML	0.871 ± 0.012	0.537 ± 0.029	0.526 ± 0.032	-
LML	0.728 ± 0.032	0.288 ± 0.037	0.390 ± 0.025	-
FedAvg	0.859 ± 0.011	0.481 ± 0.038	0.447 ± 0.062	19.2 ± 2.0
FedProx	0.859 ± 0.013	0.492 ± 0.027	0.405 ± 0.052	19.6 ± 3.8
FedAdam	0.857 ± 0.013	0.495 ± 0.032	0.452 ± 0.048	19.1 ± 2.6
FedAdagrad	0.857 ± 0.013	0.492 ± 0.034	0.463 ± 0.054	19.3 ± 1.5
FedYogi	0.854 ± 0.012	0.501 ± 0.036	**0.477 ± 0.038**	19.2 ± 3.2
FedAvgM	**0.860 ± 0.011**	**0.503 ± 0.026**	0.463 ± 0.052	20.2 ± 3.2

Table 9.5. *Test Performance with 'extreme' hospital X2*

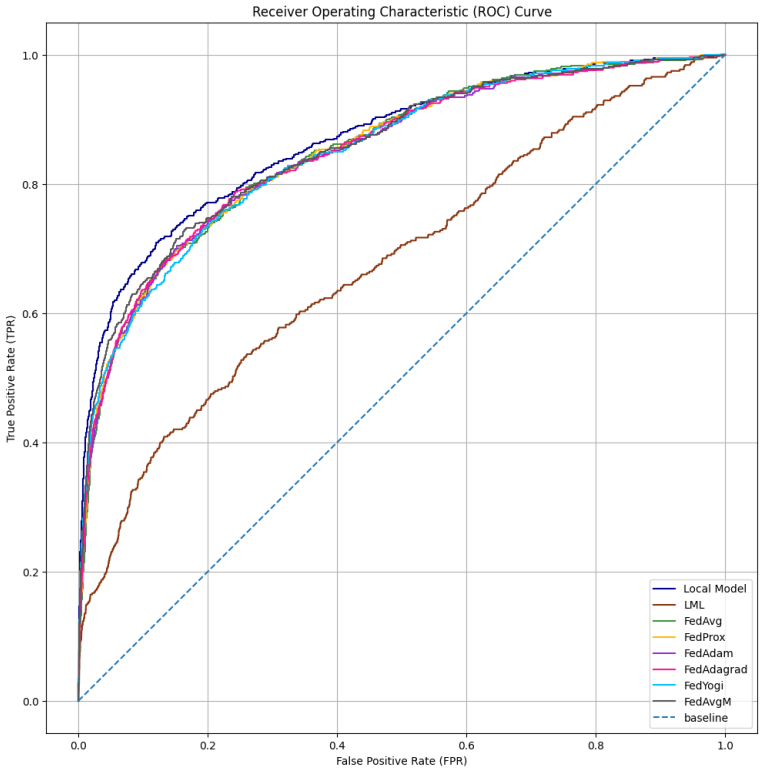


Figure 9.9. *Comparison of ROC curves obtained with CML, LML and FL approaches with the 'extreme' hospital X2*

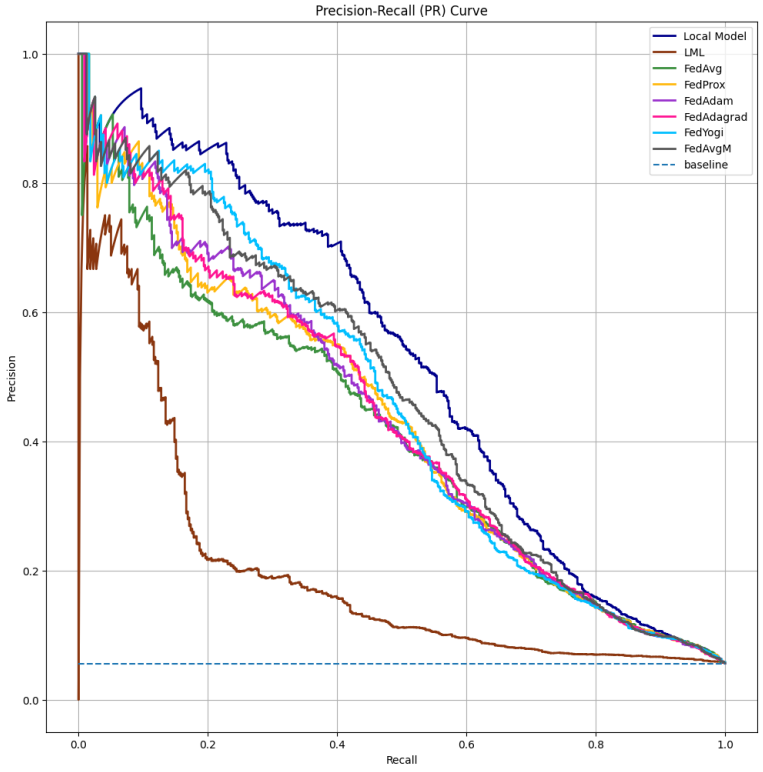


Figure 9.10. *Comparison of Precision-Recall curves obtained with CML, LML and FL approaches with 'extreme' hospital X2*

9.3 Results for Experimental Scenario 3

9.3.1 Examination of Hospital G

Table 9.6 contains the metric scores for the LML and FL approaches with and without the participation of *hospital G* in the FL training process on its local test set. It is important to restate that hospital G consists of only 930 ICU stays, showing a 5.6% mortality ratio. The local model of hospital G achieved an AUROC of 0.775, an AUPRC of 0.322 and a F1-Score of 0.320, unable to learn more complex relationships about its data due to the limited size of the training dataset. The FedAvg algorithm with the training hospitals besides hospital G (TransferFL) achieved an AUROC of 0.827, an AUPRC of 0.282 and a F1-Score of 0.341, improving in terms of AUROC and F1-Score upon the local hospital G model, but still showing limitations, which can be pinpointed on the AUPRC metric score. Nevertheless, all FL methods that included hospital G in the training process performed better than TransferFL. FedYogi was particularly effective, obtaining the best scores for every evaluation metric, with an AUROC of 0.888, an AUPRC of 0.497 and a F1-Score of 0.488. Regarding AUROC, the other FL methods reported scores ranging from 0.885 to 0.887, while, in terms of AUPRC, FedProx reached the second highest score (0.494), followed by FedAvg and FedAvgM (0.492). In relation to F1-Score, besides FedProx (0.444) and FedAvgM (0.414), the other FL algorithms reached more similar scores to FedYogi, ranging from 0.465 to 0.470. It is evident that the participation of hospital G in the federated network had a significant positive impact on the final model, as it performs considerably better than both the corresponding FL model without hospital G and the local model of hospital G.

TransferFL achieved convergence slower than the FL algorithms with the participation of hospital G. It required 17 FL rounds on average for achieving the best F1-Score on the validation set, while the other FL models required from 13.6 (FedAvgM) to 15.3 (FedProx) FL rounds on average. Overall, hospital G provided useful information to the FL network, thus helping in developing a more robust and efficient predictive model for its local data as well. It outperforms the FL model without hospital G, thus providing strong motivation for its participation in FL training.

Method	AUROC	AUPRC	F1-Score	Best FL Round
LML	0.775 ± 0.055	0.322 ± 0.156	0.320 ± 0.117	-
TransferFL	0.827 ± 0.011	0.282 ± 0.057	0.341 ± 0.033	17.0 ± 1.2
FedAvg	0.887 ± 0.036	0.492 ± 0.129	0.470 ± 0.138	14.7 ± 1.1
FedProx	0.887 ± 0.037	0.494 ± 0.130	0.444 ± 0.084	15.3 ± 1.3
FedAdam	0.886 ± 0.036	0.488 ± 0.126	0.465 ± 0.095	14.0 ± 0.8
FedAdagrad	0.885 ± 0.038	0.488 ± 0.131	0.466 ± 0.159	14.6 ± 1.1
FedYogi	**0.888 ± 0.039**	**0.497 ± 0.135**	**0.488 ± 0.142**	14.4 ± 1.0
FedAvgM	0.886 ± 0.034	0.492 ± 0.129	0.414 ± 0.073	13.6 ± 1.4

Table 9.6. *Test Performance on Hospital G*

9.3.2 Examination of Hospital C

Table 9.7 contains the metric scores for the LML and FL approaches with and without the participation of *hospital C* in the FL training process on its local test set. Hospital C contains the most data samples among the training hospitals, with 1,788 patient ICU stays and a 4.1% mortality ratio. The local model of hospital C achieved an AUROC of 0.792, an AUPRC of 0.324 and a F1-Score of 0.328. On the other hand, the FL model with the collaboration of the other 7 training hospitals, referred as TransferFL, reached an AUROC score of 0.802, an AUPRC score of 0.244 and a F1-Score of 0.320. However, both approaches are outperformed by the FL models with the training cohort of all 8 hospitals. FedYogi achieves the best AUROC score of 0.814, while the other FL algorithms reported scores in the range of 0.810 to 0.812. In terms of AUPRC, another server-side optimization FL algorithm, FedAdam, achieved the best score of 0.382, followed by FedAvgM and FedYogi with scores 0.378 and 0.376 respectively. Then, regarding F1-Score, FedAdagrad reached a score of 0.388, the best among the implemented methods in this experiment. FedAvg and FedAvgM provided the best F1-Scores besides FedAdagrad, 0.382 and 0.379 respectively, and FedAdam obtained the worst F1-Score among these FL algorithms (0.347). Once again, the local model of hospital C and the FL model trained by the remaining 7 hospitals fail to perform as good as the FL models that were developed with the participation of hospital C in the collaborative training process.

Regarding the FL round when the best F1-Score on the validation set was observed, FedAvgM required 19.3 training rounds on average, more than any other FL method. Then, TransferFL and FedAdagrad needed 17.9 FL rounds on average before obtaining the best F1-Score on the validation set, while, for the remaining FL algorithms, the training rounds before achieving this ranged from 14.8 (FedAvg) to 16.9 (FedYogi). Overall, hospital C, as the hospital with the most data within the training cohort, should theoretically be capable to train a powerful predictive model on its own, reducing the positive impact of FL on its local dataset. However, we observe that the FL methods, with the participation of hospital C, also led to a considerable increase in the evaluation metric scores for the local test set of hospital C.

Method	AUROC	AUPRC	F1-Score	Best FL Round
LML	0.792 ± 0.056	0.324 ± 0.154	0.328 ± 0.049	-
TransferFL	0.802 ± 0.033	0.244 ± 0.026	0.320 ± 0.026	17.9 ± 1.6
FedAvg	0.811 ± 0.058	0.371 ± 0.124	0.382 ± 0.136	14.8 ± 1.2
FedProx	0.810 ± 0.060	0.373 ± 0.132	0.367 ± 0.142	16.0 ± 1.4
FedAdam	0.812 ± 0.058	**0.382 ± 0.135**	0.347 ± 0.119	15.7 ± 0.7
FedAdagrad	0.811 ± 0.061	0.370 ± 0.140	**0.388 ± 0.132**	17.9 ± 3.6
FedYogi	**0.814 ± 0.060**	0.376 ± 0.140	0.354 ± 0.134	16.9 ± 0.6
FedAvgM	0.812 ± 0.059	0.378 ± 0.134	0.379 ± 0.140	19.3 ± 1.1

Table 9.7. *Test Performance on Hospital C*

9.4 Discussion

In this work, we studied the performance of federated learning on task of early prediction of ICU mortality risk, using multivariate time series data, with a real world, multicenter database. We simulated various FL environments, each of them designed to gain insight into the efficiency of FL algorithms in the presence of statistical heterogeneity. We integrated different RNN-based models into the federated learning setting, while testing the generalizability of the FL models on a foreign test set. Then, we explored how 'extreme' FL clients, that increase the heterogeneity of the federated network, affect the training of FL models in comparison to the ideal centralized ML approach. Finally, we focus on single hospital units and evaluate the performance of models created collaboratively between them and a cohort of training hospitals in comparison to other privacy-preserving approaches (a local model of the studied hospital or a FL model without its participation). The results of our experiments, which we previously presented, indicate the advantages of federated learning over the standard privacy-preserving approach of localized models, while its performance often approximates the performance of the ideal, but heavily restricted due to data privacy risks, CML approach.

9.4.1 Experimental Scenario 1

One of the main objectives of the first scenario was to assess the performance of different recurrent neural network architectures (RNN, LSTM and GRU) on this problem. Studying tables 9.1, 9.2 and 9.3, we observe that the RNN-based model produced the worst performance scores for every evaluation metric, regardless of the learning approach that was followed. However, we expected this type of predictive behavior, as the basic RNN architecture involves significantly less trainable parameters than the LSTM and GRU-based models, as shown in Figure 8.4, therefore being unable to understand more complex relationships from the input data. Then, the LSTM-based model performed significantly better, learning underlying patterns that allowed it to achieve higher scores with the CML, LML and FL approaches. Nevertheless, the less resource-intensive GRU-based model obtained the best metric scores among the 3 different RNN architectures, with all 3 approaches (the only exception is the AUROC metric with the FL approach, since the best federated learning LSTM-based model achieved a score of 0.899, slightly better than the best federated learning GRU-based model with a score of 0.892 respectively).

Another aspect of FL training that should be taken into account is the communication overhead. In our experiments, we utilize an early stopping mechanism, monitoring the F1-Score on the validation set. Consequently, faster convergence to the best F1-Score on the validation set would reduce the required FL training rounds for developing a robust model, therefore reducing the total transmission of model parameters over the network. The FL algorithms employed in this thesis, needed approximately 10 FL rounds before obtaining their best F1-Score on the validation set during training with the RNN model, Respectively, when training the LSTM model, the FL methods required just above 16.5 rounds on average, while the training of the GRU model needed less than 15 training

rounds. Considering the number of trainable parameters for each model, the number of clients and the FL training process, we calculate that training the LSTM-based model induced the heavier communication overhead (transmission of over 3.5 million trainable parameters in total), while the GRU-based model was approximately 30% lighter (with the transmission of less than 2.5 million parameters in total) and the RNN-based model was the most resource-efficient among the three, with approximately a fifth of the communication overhead of the LSTM model(with the transmission of roughly 650 thousand parameters in total).

Then, it is important to evaluate the effectiveness of federated learning, by comparing the performance of FL models with the CML and LML approaches, which constitute our two points of reference as the ideal and baseline performance respectively. Table 9.8 shows the performance gap between the best FL approach and the CML and LML approaches for each model architecture and evaluation metric, thus giving us an insight of how well the FL models approximate the ideal CML and build upon the baseline LML.

Metric	RNN		LSTM		GRU	
	FL-CML	FL-LML	FL-CML	FL-LML	FL-CML	FL-LML
AUROC	+0.002	+0.074	+0.005	+0.096	−0.003	+0.085
AUPRC	−0.036	+0.125	−0.023	+0.150	−0.032	+0.147
F1-Score	−0.137	+0.083	−0.051	+0.054	−0.029	+0.099

Table 9.8. *Performance analysis of the best FL model of each RNN architecture, in comparison to the CML and LML*

We observe that the LSTM-based models, trained within a federated learning environment, perform more similarly to the respective CML model, in terms of AUROC (+0.005) and AUPRC (-0.023), and indicate a larger improvement in performance over the baseline LML models (+0.096 and +0.150 respectively). Regarding F1-Score, the GRU-based model seems to be the one better adapted to the FL setting, scoring only 0.029 less than the CML approach and a substantial 0.099 more than the LML approach. These observations, along with the communication overhead analysis, render GRU architecture the most suitable for the task of early prediction of ICU mortality risk using MTS data, within the FL framework we conduct our experiments.

All models were evaluated on a foreign test set, consisting of data from 4 hospitals that did not participate in the training phase. This allows us to draw conclusions about the performance of the model, with regard to its predictions, and the model's ability to generalize beyond the training hospitals. We observe that, despite the non-IID local datasets of the FL clients, the FL models achieve noteworthy metric scores, closely resembling the performance of the CML approach which has access to the entire global dataset and optimizes the training process. We observe that, based on the performance on each evaluation metric, the standard FedAvg algorithm performs considerably well, especially in combination with the RNN-based and LSTM-based models. This highlights the inherent robustness of the simplest and most commonly used FL algorithm in an

environment with a certain degree of heterogeneity. The FedProx algorithm, which regulates the client updates with a proximal term, outperformed FedAvg for certain metrics and model architectures, such as the F1-Score of the LSTM model (0.438 over 0.417) and the AUPRC score of the GRU model (0.507 over 0.499), highlighting its ability to handle statistical heterogeneity more efficiently. The server-side optimization FL algorithms, namely the 3 FedOpt algorithms and FedAvgM, generally converged faster than FedAvg and FedProx when training the models with the RNN and LSTM architectures, as they incorporate additional optimization techniques, usually resulting in increased training efficiency. However, this may have led them to slightly underfitting during training with the RNN and LSTM models, performing worse than FedAvg and FedProx on average. Considering the GRU architecture experiment, FedAdam and FedAdagrad required more FL rounds than FedAvg and FedProx before convergence, which translated into an improvement in their predictive performance, with FedAdagrad providing the best AUROC and F1-Score among the FL algorithms on this experiment. Taking everything into account, a more exhaustive fine-tuning could potentially lead to some FL algorithms performing even better under certain circumstances, however our results indicate that federated learning is robust to heterogeneous federated networks, leading to the development of powerful, generalizable models, in contrast to the privacy-preserving localized ML.

9.4.2 Experimental Scenario 2

This second scenario investigated the performance of federated learning in the presence of 'extremely' divergent FL clients, in terms of dataset size and class distribution, from the cohort of training hospitals. First, we added hospital X1 to the training cohort, which contains 1,900 ICU stays, with just 10 of them resulting in mortality, a mere 0.5%. Due to its size, this hospital has a contribution of 17% during the aggregation of local model parameters, significantly affecting the final FL model. As the test set for this experiment derives from the training hospitals and is characterized by a mortality ratio of 4.3%, we would want to regulate the influence of this hospital, while simultaneously developing a robust global model.

At first glance, it seems that the federated learning algorithms perform considerably well, nevertheless affected by the addition of hospital X1. One of our points of focus is FedProx, which surpasses the standard FedAvg in terms of AUROC and F1-Score and is slightly behind regarding AUPRC. Even for AUPRC, Figure 9.8 shows that FedAvg achieves a better combination of precision and recall only as recall is substantially reduced, roughly approximating 0. This is an indicator of the effect of the proximal term on the client updates, as they tend to remain closer to the global model of the previous FL round. Even though increasing the proximal term could further restrict the divergent behavior of hospital X1, it could deteriorate the FL system's ability to capture underlying patterns and relationships from the other FL clients. On a similar note, the server-side optimization FL algorithms displayed worse performance in these highly heterogeneous settings (in terms of AUROC and AUPRC), with the exception of FedYogi and FedAdagrad, which provide the best F1-Scores. This highlights a weakness of this category of algorithms to effectively

handle this type of client heterogeneity. As a result, we deduce that, in heterogeneous federated networks with highly influential 'extreme' clients, local-side mechanisms are more suitable to regulate local model updates, such as a proximal term with FedProx or control variates with SCAFFOLD. However, the FL approach is seriously impacted, in comparison to CML. As shown in Table 9.9, the best FL metric scores diverge considerably more from the respective CML scores, in the presence of hospital X1. Even though the FL performance is significantly better than the performance of LML models (which also include hospital X1), we evaluate federated learning with CML as a point of reference, therefore regarding this scenario demanding for federated learning.

Analogously, our second sub-scenario involves hospital X2, which contains only 300 ICU stays, with 75 of them resulting in mortality, a ratio of 25%. With regard to its size, hospital X2 has a small contribution of 3% during the aggregation of the local model updates. Once more, the dataset of this artificial hospital is biased in comparison to the other participating hospitals and the global data distribution, requiring special treatment during FL training, while the combined test set from the participating hospitals is characterized by a mortality ratio of 5.7%.

The results for this experiment provide us with an insight into the performance of the FL algorithms from another point of view. The FedProx algorithm could be considered the most unsuitable among the FL optimization algorithms for this sub-scenario. In this case, the regulation of the local model updates with the proximal term does not allow hospital X2 to train 'freely' its local model. As a result, the minor contribution of hospital X2 to the final model is actually reduced even further, since the hospital X2 model update is limited to be closer to the previous global model. This can be observed both with AUPRC and F1-Score, where especially the value of the latter for FedProx is considerably lower than the other FL methods. Similarly, we observe that the standard FedAvg algorithm is outperformed by the server-side optimization FL algorithms, as it also fails to effectively utilize the available information from the heterogeneous network of clients. On the contrary, the 3 FedOpt algorithms (especially FedYogi) and FedAvgM perform significantly better, in relation to AUPRC and F1-Score which are the most informative metrics for this imbalanced task. This indicates that server-side optimization and local-side training without regulations is a more appropriate FL approach to this problem, as it does not further weigh down the influence of clients with smaller datasets. Table 9.9 shows that the difference between the best FL and CML scores has increased, however not as much as in the first sub-scenario, partly because of the size of hospital X2.

Metric	GRU		GRU + X1		GRU + X2	
	FL-CML	FL-LML	FL-CML	FL-LML	FL-CML	FL-LML
AUROC	−0.003	+0.085	−0.016	+0.165	−0.011	+0.132
AUPRC	−0.032	+0.147	−0.056	+0.220	−0.034	+0.215
F1-Score	−0.029	+0.099	−0.065	+0.144	−0.049	+0.087

Table 9.9. *Performance analysis of the best FL models from scenario 2 and the GRU-based model from scenario 1, in comparison to the CML and LML*

9.4.3 Experimental Scenario 3

The third scenario focuses on individual hospitals, evaluating the impact of their participation in a FL environment on their local test sets. Our first point of interest is hospital G of the training cohort, containing 930 ICU stays with a 5.6% mortality ratio. Table 9.6 indicates that hospital G contains few data to train a robust predictive model and the privacy-preserving transfer learning alternative, adopting a FL model trained on the remaining hospitals of the training cohort, fails to generalize sufficiently to capture the characteristics of the hospital G's dataset. By incorporating hospital G into the federated training network, we observe a substantial improvement in every metric score, regardless of the employed FL method. Even though hospital G has a minor 8% contribution during the model weights aggregation at the server-side, it clearly improves the final model by exposing it to a subset of its local data during training. FedYogi algorithm achieved the best scores in every evaluation metric, however most FL algorithms had roughly similar performance in terms of AUROC and AUPRC. As we expected, by participating in FL training, hospital G advances the development of a more generalizable model, which simultaneously constitutes the most effective, privacy-preserving solution for predictions on its local data.

Since hospital G contains limited data, it is strongly motivated to participate in a FL environment, in order to develop a robust model that has also been trained on its unique set of patient ICU stays. Our next step was to explore the impact of participation for a larger hospital, thus focusing on hospital C. It contains data for 1,788 ICU stays with a 4.1% mortality ratio, the largest hospital of our training cohort. According to our results, shown in Table 9.7, the local model of hospital C achieves slightly better metric scores than those achieved earlier by the local model of hospital G, mainly due to the increase in data availability. The transfer learning FL model performs overall worse than the local model, potentially due to the absence of the hospital C that provided the most data to the federated network (which also changes the dynamics of the FL environment, with all FL clients now contributing more to the final model). Then, we observe that the FL methods with the participation of hospital C achieve the best scores for all evaluation metrics, even though they do not improve upon the performance of the local model as much as for hospital G. One of the main reason behind this behavior is that hospital C already has a sufficient amount of data to train a satisfactory predictive model. Nevertheless, participating in the development of a FL model again leads to a noticeable improvement, especially in terms of AUPRC and F1-Score. Consequently, even though a larger hospital seems to gain less from FL participation, in contrast to a smaller-sized hospital, there are still significant advantages to be reaped from collaboratively building a FL model, with regard to both improving performance on the local data and obtaining a generalizable model with respect to the global data.

Chapter 10

Epilogue

This thesis addressed the problem of multi-center collaboration on the early ICU mortality risk prediction problem with multivariate time series data. We studied the effectiveness of different privacy-preserving FL algorithms on a realistic, non-IID federated network of hospitals, comparing them to the ideal centralized ML approach and the baseline, privacy-preserving local ML approach. Our study cohort derives from the eICU Collaborative Research Database, which contains real world data for ICU stays from multiple hospitals across the US, thus allowing us to recreate a real-life, heterogeneous environment. To train our models, we employed deep neural network architectures, focusing on RNN, LSTM and GRU, and their integration within the FL framework. We designed a series of experimental scenarios in order to gain insight into different aspects of federated learning, such as global model generalizability, sensitivity to 'extreme' FL clients and the importance of participation in the FL setting.

The results of our experiments showcase the potential of federated learning in a task where data privacy and security is of paramount importance. Among the three recurrent neural network models we implemented, we observed that the GRU-based model constitutes the golden ratio in the tradeoff between performance and communication overhead in the FL setting for this task, whereas the LSTM-based model was the most resource-intensive and the performance of the RNN-based model deviated the most from the ideal CML method. Simultaneously, the FL models achieved robust scores on a foreign test set, therefore highlighting the capacity of FL models to generalize beyond the training data in contrast to localized models. Then, in the presence of a FL client with an 'extreme' data distribution against the under-represented class and significant contribution to the model aggregation phase, our results indicate local-side mechanisms, such as the proximal term of FedProx, are more suitable to regulate any negative effects on the final model. On the other hand, in the presence of a FL client with an 'extreme' data distribution towards the under-represented class (in comparison to the training cohort) and minor contribution to the global model, we observe that server-side optimization FL algorithms (such as FedYogi) perform better, as they allow the FL clients to train without restrictions and employ optimization techniques during weight aggregation. Moreover, we measure the impact of a hospital partaking in a FL training process, showing that, besides smaller sized hospitals, larger sized, more 'powerful' hospitals can also be individually benefited from contributing to the development of a FL model.

However, we should state certain limitations of our experiments, which point out some intriguing future research directions. Initially, regarding the task of early ICU mortality risk prediction, it would be interesting to develop ML models that process both temporal variables, such as vital signs and lab tests, but also static characteristics, such as age, diagnosis and patient history, and integrate them into the FL setting, measuring their performance. Another interesting direction would be to explore the re-sampling frequency of the MTS data used in this thesis and determine the frequency that provides the most useful information to the predictive models. Then, regarding the federated learning setting, experiments with a larger number of participating hospitals would create different dynamics in the FL environment and provide more representative results of the study cohort. Moreover, one could focus on specific FL algorithms and study their behavior in different settings with a more exhaustive hyperparameter fine-tuning, as well as develop personalized federated learning algorithms to adapt more effectively to individual client datasets.

Bibliography

[1] *eICU Collaborative Research Database Code Repository*. https://github.com/MIT-LCP/eicu-code. Access date: 5-3-2023.

[2] Aston Zhang, Zachary Lipton, Mu Li και Alexander Smola. *Dive into Deep Learning*. 2021.

[3] *What Is Federated Learning?* https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/. Access date: 10-5-2023.

[4] Robin John Hyndman και George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2η έκδοση, 2018.

[5] *Time series in healthcare: challenges and solutions*. https://www.vanderschaar-lab.com/time-series-in-healthcare/. Access date: 5-3-2023.

[6] Fardin Ghorbani, Javad Shabanpour, Sina Beyraghi, Hossein Soleimani, Homayoon Oraizi και M. Soleimani. *A deep learning approach for inverse design of the metasurface for dual-polarized waves*. *Applied Physics A*, 127:869, 2021.

[7] *Why are deep neural networks hard to train?* http://neuralnetworksanddeeplearning.com/chap5.html. Access date: 29-4-2023.

[8] *What is the Convolutional Neural Network Architecture?* https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/. Access date: 3-5-2023.

[9] Pradeep Hewage, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri και Yonghuai Liu. *Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station*. *Soft Comput.*, 24(21):16453–16482, 2020.

[10] *An Intuitive Explanation of LSTM*. https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c. Access date: 5-5-2023.

[11] Sheng Shen, Tianqing Zhu, Di Wu, Wei Wang και Wanlei Zhou. *From distributed machine learning to federated learning: In the view of data privacy and security*. *Concurrency and Computation: Practice and Experience*, 34, 2020.

[12] *Federated Learning: The Shift from Centralized to Distributed On-Device Model Training*. https://www.altexsoft.com/blog/federated-learning/. Access date: 10-5-2023.

[13] Mina Aghaei Dinani. *Federated learning for vehicle trajectory prediction*. Μεταπτυχιακή διπλωματική εργασία, POLITECNICO DI TORINO, 2020.

[14] Trung Kien Dang, Xiang Lan, Jianshu Weng και Mengling Feng. *Federated Learning for Electronic Health Records*. *ACM Trans. Intell. Syst. Technol.*, 13(5), 2022.

[15] Korbinian Randl, Núria Armengol, Lena Mondrejevski και Ioanna Miliou. *Early prediction of the risk of ICU mortality with Deep Federated Learning*, 2022.

[16] *General Data Protection Regulation GDPR*. `https://gdpr-info.eu/`. Access date: 5-5-2023.

[17] Tom Joseph Pollard, Alistair Edward William Johnson, Jesse Raffa και Omar Badawi. *The eICU Collaborative Research Database*, 2017.

[18] Peter Zhang. *Zhang, G.P.: Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model. Neurocomputing 50, 159-175*. *Neurocomputing*, 50:159–175, 2003.

[19] E.L. Wike. *Data Analysis: A Statistical Primer for Psychology Students*. Routledge, 1η έκδοση.

[20] Robert Andersen. *Modern Methods for Robust Regression*. 2007.

[21] Isabelle Guyon και André Elisseeff. *An Introduction to Variable and Feature Selection*. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[22] Jeff Heaton. *An empirical analysis of feature engineering for predictive modeling*. *SoutheastCon 2016*, σελίδες 1–6, 2016.

[23] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1η έκδοση, 1997.

[24] Simon S. Haykin. *Neural Networks and Learning Machines*. 2010.

[25] Yann LeCun, Y. Bengio και Geoffrey Hinton. *Deep Learning*. *Nature*, 521:436–44, 2015.

[26] Ahmed Tealab. *Time series forecasting using artificial neural networks methodologies: A systematic review*. *Futur. Comput. Inform. J.*, 3(2):334–340, 2018.

[27] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson και Blaise Agüeray Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*, 2023.

[28] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo,

Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu και Sen Zhao. *Advances and Open Problems in Federated Learning*, 2021.

[29] Reza Shokri και Vitaly Shmatikov. *Privacy-Preserving Deep Learning*. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, σελίδα 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery.

[30] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar και Virginia Smith. *Federated Optimization in Heterogeneous Networks*, 2020.

[31] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar και H. Brendan McMahan. *Adaptive Federated Optimization*, 2021.

[32] Tzu Ming Harry Hsu, Hang Qi και Matthew Brown. *Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification*, 2019.

[33] Tzu Ming Harry Hsu, Hang Qi και Matthew Brown. *Federated Visual Classification with Real-World Data Distribution*, 2020.

[34] William Knaus, D Wagner, EA Draper, J Zimmerman, M Bergner, P Bastos, CA Sirio, D Murphy, T Lotring και AM Damiano. *The APACHE III prognostic system: Risk prediction of hospital mortality for critically III hospitalized adults*. *Chest*, 100:1619–36, 1992.

[35] Philipp Metnitz, Rui Moreno, Eduardo Almeida, Barbara Jordan, Peter Bauer, Ricardo Campos, Gaetano Iapichino, David Edbrooke, Maurizia Capuzzo και Jean Roger Gall. *SAPS 3 - From evaluation of the patient to evaluation of the intensive care unit. Part 1: Objectives, methods and cohort description*. *Intensive care medicine*, 31:1336–44, 2005.

[36] Alistair E W Johnson, Tom J Pollard, Lu Shen, Li Wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi και Roger G Mark. *MIMIC-III, a freely accessible critical care database*. *Sci. Data*, 3(1):160035, 2016.

[37] Lena Mondrejevski, Ioanna Miliou, Annaclaudia Montanino, David Pitts, Jaakko Hollmén και Panagiotis Papapetrou. *FLICU: A Federated Learning Workflow for Intensive Care Unit Mortality Prediction*. *2022 IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS)*, σελίδες 32–37, 2022.

[38] Takaya Saito και Marc Rehmsmeier. *The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets*. *PLOS ONE*, 10(3):1–21, 2015.

[39] *What is time series data?* `https://www.influxdata.com/what-is-time-series-data/`. Access date: 5-3-2023.

[40] Ratnadip Adhikari και R. Agrawal. *An Introductory Study on Time series Modeling and Forecasting*. 2013.

[41] Robert Shumway και David Stoffer. *Time Series Analysis and Its Applications With R Examples*, τόμος 9. 2011.

[42] Chris Chatfield. *The Analysis of Time Series: An Introduction, Sixth Edition*. 2016.

[43] Ian Jolliffe και Jorge Cadima. *Principal component analysis: A review and recent developments*. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374:20150202, 2016.

[44] Helmut Luetkepohl. *The New Introduction to Multiple Time Series Analysis*. 2005.

[45] KarlØyvind Mikalsen, Filippo Maria Bianchi, Cristina Soguero-Ruiz και Robert Jenssen. *Time series cluster kernel for learning similarities between multivariate time series with missing data*. *Pattern Recognition*, 76:569–581, 2018.

[46] Peter J Brockwell και Richard A Davis. *Introduction to Time Series and Forecasting*. Springer, New York, 3η έκδοση, 2002.

[47] Estela Dagum και Silvia Bianconcini. *Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation*. 2016.

[48] Robert B. Cleveland, William S. Cleveland, Jean E. McRae και Irma Terpenning. *STL: A Seasonal-Trend Decomposition Procedure Based on Loess (with Discussion)*. *Journal of Official Statistics*, 6:3–73, 1990.

[49] Senthamarai Kaliyaperumal, Manoj Kuppusamy και S. Arumugam. *Labeling Methods for Identifying Outliers*. *International Journal of Statistics and Systems*, 10:231–238, 2015.

[50] Peter J. Huber. *Robust Estimation of a Location Parameter*. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[51] Oded Maimon και Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag, Berlin, Heidelberg, 2005.

[52] Mark Hall. *Correlation-Based Feature Selection for Machine Learning*. *Department of Computer Science*, 19, 2000.

[53] Dongkuan Xu και Yingjie Tian. *A Comprehensive Survey of Clustering Algorithms*. *Annals of Data Science*, 2, 2015.

[54] William McGinnis, Chapman Siu, Andre S και Hanyu Huang. *Category Encoders: a scikit-learn-contrib package of transformers for encoding categorical data*. *The Journal of Open Source Software*, 3:501, 2018.

[55] Adam Coates, Andrew Ng και Honglak Lee. *An Analysis of Single-Layer Networks in Unsupervised Feature Learning*. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011.

[56] Sergey Ioffe και Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015.

[57] Shin Fu Wu, Chia Yung Chang και Shie Jue Lee. *Time Series Forecasting with Missing Values*. τόμος 1, 2015.

[58] Scott Zeger, Rafael Irizarry και Roger Peng. *On Time Series Analysis of Public Health and Biomedical Data*. *Annual review of public health*, 27:57–79, 2006.

[59] Zitao Liu και Milos Hauskrecht. *A Personalized Predictive Framework for Multivariate Clinical Time Series via Adaptive Model Selection*. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, σελίδα 1169–1177, New York, NY, USA, 2017. Association for Computing Machinery.

[60] Omar Enzo Santangelo, Vito Gentile, Stefano Pizzo, Domiziana Giordano και Fabrizio Cedrone. *Machine Learning and Prediction of Infectious Diseases: A Systematic Review*. *Machine Learning and Knowledge Extraction*, 5(1):175–198, 2023.

[61] Stuart Russell και Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3η έκδοση, 2009.

[62] Mehryar Mohri, Afshin Rostamizadeh και Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[63] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[64] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[65] Jürgen Schmidhuber. *Deep learning in neural networks: An overview*. *Neural Networks*, 61:85–117, 2015.

[66] Andreas Zell. *Simulation neuronaler Netze*. 1994.

[67] Marvin Minsky και Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 2017.

[68] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard και L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition*. *Neural Computation*, 1(4):541–551, 1989.

[69] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj και Alexandros Iosifidis. *Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks*. *2017 IEEE 19th Conference on Business Informatics (CBI)*, τόμος 01, σελίδες 7–12, 2017.

[70] Hamed Aghdam και Elnaz Heravi. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. 2017.

[71] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter και Gregory D. Hager. *Temporal Convolutional Networks for Action Segmentation and Detection*. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 1003–1012, 2017.

[72] Jining Yan, Lin Mu, Lizhe Wang, Rajiv Ranjan και Albert Y Zomaya. *Temporal convolutional networks for the advance prediction of ENSO*. *Sci. Rep.*, 10(1):8055, 2020.

[73] Fisher Yu και Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*, 2016.

[74] Pedro Lara-Benítez, Manuel Carranza-García, José M. Luna-Romera και José C. Riquelme. *Temporal Convolutional Networks Applied to Energy-Related Time Series Forecasting*. *Applied Sciences*, 10(7):2322, 2020.

[75] Sepp Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. 1991.

[76] Jürgen Schmidhuber. *The 2010s: Our Decade of Deep Learning / Outlook on the 2020s*. `https://people.idsia.ch/~juergen/2010s-our-decade-of-deep-learning.html`. Access date: 5-5-2023.

[77] Kyunghyun Cho, Bartvan Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk και Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, 2014.

[78] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho και Yoshua Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, 2014.

[79] M. Schuster και K.K. Paliwal. *Bidirectional Recurrent Neural Networks*. *Trans. Sig. Proc.*, 45(11):2673–2681, 1997.

[80] *Understanding Bidirectional RNN*. `https://www.scaler.com/topics/deep-learning/bidirectional-rnn/`. Access date: 5-5-2023.

[81] Qiang Yang, Yang Liu, Tianjian Chen και Yongxin Tong. *Federated Machine Learning: Concept and Applications*. *ACM Trans. Intell. Syst. Technol.*, 10(2), 2019.

[82] Muhammad Asad, Ahmed Moustafa και Takayuki Ito. *Federated Learning Versus Classical Machine Learning: A Convergence Comparison*. 2020.

[83] *Facebook–Cambridge Analytica data scandal*. https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal. Access date: 5-5-2023.

[84] *Distributed Machine Learning Vs Federated Learning: Which Is Better?* https://analyticsindiamag.com/distributed-machine-learning-vs-federated-learning-which-is-better/. Access date: 9-5-2023.

[85] *Machine Learning: Why Scaling Matters*. https://www.codementor.io/blog/scaling-ml-6ruo1wykxf. Access date: 10-5-2023.

[86] Jakub Konečný, Brendan McMahan και Daniel Ramage. *Federated Optimization:Distributed Optimization Beyond the Datacenter*, 2015.

[87] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage και Jason Roselander. *Towards Federated Learning at Scale: System Design*, 2019.

[88] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish και Ramesh Raskar. *Split learning for health: Distributed deep learning without sharing raw patient data*, 2018.

[89] Enmao Diao, Jie Ding και Vahid Tarokh. *HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients*, 2021.

[90] Shiva Raj Pokhrel και Jinho Choi. *Federated Learning With Blockchain for Autonomous Vehicles: Analysis and Design Challenges*. *IEEE Transactions on Communications*, 68(8):4734–4746, 2020.

[91] *Optimization: Stochastic Gradient Descent*. http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/. Access date: 11-5-2023.

[92] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich και Ananda Theertha Suresh. *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*, 2021.

[93] Cynthia Dwork, Frank McSherry, Kobbi Nissim και Adam Smith. *Calibrating Noise to Sensitivity in Private Data Analysis*. *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, σελίδα 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[94] Cynthia Dwork. *Differential Privacy: A Survey of Results*. τόμος 4978, σελίδες 1–19, 2008.

[95] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar και Li Zhang. *Deep Learning with Differential Privacy*. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[96] Andrew C. Yao. *Protocols for Secure Computations*. *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, σελίδα 160–164, USA, 1982. IEEE Computer Society.

[97] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal και Karn Seth. *Practical Secure Aggregation for Federated Learning on User-Held Data*, 2016.

[98] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal και Karn Seth. *Practical Secure Aggregation for Privacy-Preserving Machine Learning*. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, σελίδα 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[99] Andrew Fang, Wan Lim και Tharmmambal Balakrishnan. *Early warning score validation methodologies and performance metrics: A systematic review*, 2020.

[100] Alistair E W Johnson και Roger G Mark. *Real-time mortality prediction in the Intensive Care Unit*. *AMIA Annu. Symp. Proc.*, 2017:994–1003, 2017.

[101] Sanjay Purushotham, Chuizheng Meng, Zhengping Che και Yan Liu. *Benchmark of Deep Learning Models on Large Healthcare MIMIC Datasets*, 2017.

[102] Aya Awad, Mohamed Bader-El-Den, James McNicholas, Jim Briggs και Yasser Elsonbaty. *Predicting hospital mortality for intensive care unit patients: Time-series analysis*. *Health Informatics Journal*, 26:146045821985032, 2019.

[103] Joon Lee, Daniel J. Scott, Mauricio Villarroel, Gari D. Clifford, Mohammed Saeed και Roger G. Mark. *Open-access MIMIC-II database for intensive care research*. *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, σελίδες 8315–8318, 2011.

[104] Thanakron Na Pattalung, Thammasin Ingviya και Sitthichok Chaichulee. *Feature Explanations in Recurrent Neural Networks for Predicting Risk of Mortality in Intensive Care Patients*. *Journal of Personalized Medicine*, 11, 2021.

[105] Scott Lundberg, Bala Nair, Monica Vavilala, Mayumi Horibe, Michael Eisses, Trevor Adams, David Liston, Daniel Low, Shu Fang Newman, Jerry Kim και Su In Lee. *Explainable machine-learning predictions for the prevention of hypoxaemia during surgery*. *Nature Biomedical Engineering*, 2, 2018.

[106] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi και Roger Mark. *MIMIC-IV*, 2023.

[107] Geun Hyeong Lee και Soo Yong Shin. *Federated learning on clinical benchmark data: Performance assessment*. *J. Med. Internet Res.*, 22(10):e20891, 2020.

[108] Andrius Budrionis, Magda Miara, Piotr Miara, Szymon Wilk και Johan Gustav Bellika. *Benchmarking PySyft Federated Learning Framework on MIMIC-III Dataset*. *IEEE Access*, 9:116869–116878, 2021.

[109] Li Huang, Andrew Shea, Huining Qian, Aditya Masurkar, Hao Deng και Dianbo Liu. *Patient Clustering Improves Efficiency of Federated Machine Learning to Predict Mortality and Hospital Stay Time Using Distributed Electronic Medical Records*. *Journal of Biomedical Informatics*, 99:103291, 2019.

[110] Akhil Vaid, Suraj Jaladanki, Jie Xu, Shelly Teng, Arvind Kumar, Samuel Lee, Sulaiman Somani, Ishan Paranjpe, Jessica Freitas, Tingyi Wanyan, Kipp Johnson, Mesude Bicak, Eyal Klang, Joon Young Kwon, Anthony Costa, Shan Zhao, Riccardo Miotto, Alexander Charney και Benjamin Glicksberg. *Federated Learning of Electronic Health Records to Improve Mortality Prediction in Hospitalized Patients With COVID-19: Machine Learning Approach*. 2021.

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUPRC	Area Under Precision-Recall Curve
AUROC	Area Under Receiver Operating Characteristic
BiLSTM	Bidirectional Long Short-Term Memory
BRNN	Bidirectional Recurrent Neural Network
CITI	Collaborative Institutional Training Initiative
CNN	Convolutional Neural Network
DL	Deep Learning
DML	Distributed Machine Learning
DP	Differential Privacy
DNN	Deep Neural Network
EHR	Electronic Health Records
FL	Federated Learning
FNN	Feedforward Neural Network
GDPR	General Data Protection Regulation
GRU	Gated Recurrent Unit
HIPPA	Health Insurance Portability and Accountability Act
ICU	Intensive Care Unit
IID	Independent and Identically Distributed
IoT	Internet of Things
LSTM	Long Short-Term Memory
ML	Machine Learning
MTS	Multivariate Time Series
NN	Neural Network
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
TCN	Temporal Convolutional Network