SUPPLEMENTARY MATERIAL

Anonymous authors
Paper under double-blind review

FEATURES

Tables 1 and 2 contain the patient-specific and hospital-specific features used for prediction for the ward admission problem. Table 3 contains the features used for the eICU prediction problem.

Bacteriology test requested?	Biochemical tests requested?
Blood cultures requested	Blood gas test requested?
CT scan requested?	Cardiac enzyme test requested?
Clotting study requested	Blood cross-matching requested?
Heart rate at entry	Dental investigation requested?
ECG requested?	Haematology test requested?
MRI scan requested?	Immunology test requested?
Continuous vitals requested?	No tests requested
Orthopedic tests requested?	Other tests requested?
Pregnancy test requested?	Respiratory rate at entry
Serology test requested?	Previous specialty
Toxicology test requested?	Ultrasound test requested?
Urine test requested?	X-ray scan requested?
Admission method	Admission source
Age	Experiencing atrial fibrillation?
# historic diagnoses	Previous management
Previous admission to ED?	Ethnic category
Frequently admitted?	Gender
Dist. of address to hospital	No. Investigations requested
Previous ED visit days ago	Previous visit LOS
Mortality indicator severity score	Historic diagnosis codes

Table 1: Table containing the patient specific features available at initial medical assessment that were used in the ward admission dataset.

Hours of sunlight	Hour admitted to ED
Month admitted to ED	# ED Attendees in last 12 hours
# ED Attendees in last 4 hours	# ED Attendees in last 8 hours
# ED Attendees in last hour	# Breaches last 12 hours
# Breaches last 4 hours	# Breaches last 8 hours
# Breaches last hour	Has it rained today?
Min. day temp (degrees)	Max. day temp (degrees)
Weekday (one-hot)	

Table 2: Table containing the environmental/hospital features that were used in the ward admission dataset.

LEARNING FROM MULTIPLE HOSPITALS

When training on the eICU dataset, we utilised data from individual hospitals as the separate nodes in the federated system. A natural question that arises is how the increasing the number of nodes in the system affects the final performance of the student.

Heart rate	Mean arterial pressure
Diastolic blood pressure	Systolic blood pressure
Blood oxygen saturation	Respiratory rate
Temperature	Glucose
Oxygen delivery rate	pH
Height	Weight
Age	Admission diagnosis
Ethnicity	Gender
Glasgow coma score total	

Table 3: Table containing the features used for mortality prediction on the eICU dataset. Features are extracted from the first measurement of each patient upon admission to ICU.

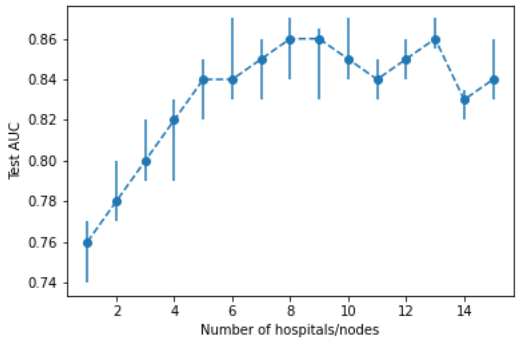

Figure 1: The number of nodes in the federated system (i.e., number of hospitals) versus the student performance at test time. Note that in this figure the seed of the scheduler is held constant and the error bars correspond to the scores generated from using different hospitals for training. So for N=k nodes, the error bars correspond to training only using data from ten different combinations of k hospitals.

In Figure 1 we carry out an ablation on the number of nodes in the system and how it affects performance. In this plot, the scheduler is held at constant seed and the error bars are generated due to selecting different hospitals to train on. There are a total of 208 hospitals in the eICU dataset, but we only select from the top 20 in terms of volume of data recorded. As a result, Figure 1 shows error bars for the difference in performance when ten different combinations of hospital data are used. We can see that with more data being used in the system, the final performance generally increases. We also see that the variance in the performance starts to decrease with the increase in the number of nodes. Given the increase in the volume of data being trained with as nodes are added, this aligns with expectations. The performance seems to plateau indicating that adding more nodes to the system may not necessarily be beneficial for performance. This could be useful for real-world application as it will allow practitioners to prioritise data centres with the highest quality data for their federated systems.

Node Selection

Figure 2 shows the scheduler outputs during training for the hospital admission problem and Figure 3 shows the performance of the student and the actions of the teacher during training.

Iterative Learning

In our approach we choose to exploit local models to select data and therefore extract gradients that we use to update a global model. The advantage of this approach is that it provides flexibility for a poisoned node within the federated system to be discounted or unused. This can also be done on the

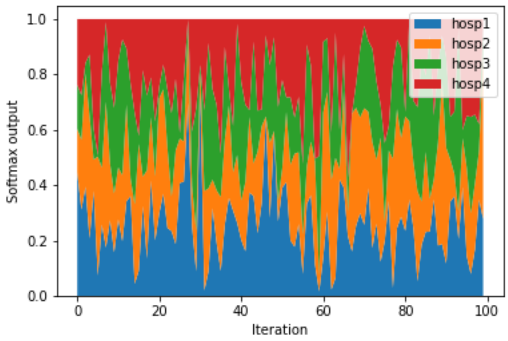

Figure 2: A four-node federated system being scheduled for training a student on the hospital admission location prediction problem. The different colours represent the different nodes.

Figure 3: The performance of the student and actions taken by the teachers (scheduled according to Figure 2) at each node to train the student. The orange 'x' and blue bar indicate the first and second outputs of the teacher respectively (index of data along the curriculum that is selected and how much data around this to include in that batch). The red line shows the performance of the student on the held-out test set on the hospital admission problem.

fly without the need to inspect the local models after each training run or the data stored at each node. In the following section we look at the ways that a node can be compromised and see how our setup may be able to avoid the global model being poisoned by these scenarios.

Compromised Local Models at Nodes

The next experiment we investigate is how our system trains when there are compromised local models (whether it is the student or the teacher). We do this by replacing one of the weights of the teacher on one of the local nodes with random values. Due to the setup, the effect of having a compromised student or teacher is the same: a high loss which encourages the scheduler to change its selection. Figure 4 shows how the scheduler selects nodes to train from. We see initially nodes 1 and 0 are used to train before the scheduler attempts to use the poisoned teacher. After repeated reductions in the federated validation sets, the scheduler rapidly changes its selection favouring nodes 4 and increasingly 3. With further training, we see this rapid removal of training using node 1 continue whenever it is encountered.

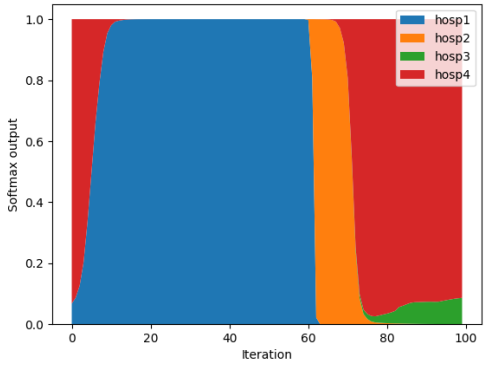

Figure 4: Scheduler selection for a four-node federated system. The hosp1 node (orange) corresponds to the poisoned teacher (randomised weights). We see that after brief selection, the scheduler reduces the contribution of this node. The student is being trained on the MNIST task.

SELECTING THE RIGHT TEACHER

In our first set of experiments we aimed to see if the scheduler would be able to select the appropriate teacher for the learning task at hand. Our scheduler's task is to select a teacher from three different nodes to train the student. We pre-trained three teachers for separate tasks (hospital admission, CIFAR-10 and MIMIC-III mortality prediction) and allowed our scheduler to choose from these in order to source the data for training. Figure 5 shows how with training, the scheduler learns to assign the teaching job to the node that contains the teacher trained to teach CIFAR-10 learning. As training progresses, the response from the scheduler becomes entirely dominated by the node in the federated system that corresponds to the appropriate teacher for the task. From this we see that our approach allows for robust teacher selection (further examples of this are included in the supplementary material).

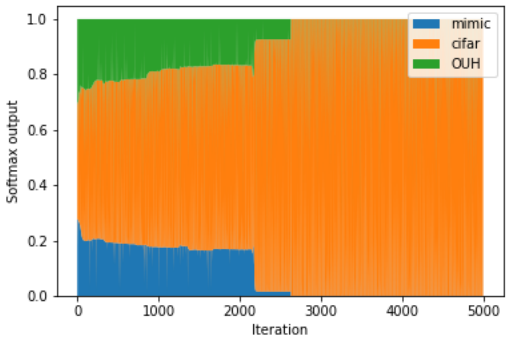

Figure 5: Scheduler selection as the student is trained. The student is being trained on CIFAR-10 and the scheduler learns to use the CIFAR-10 teacher to teach the student.

COMPROMISED DATA AT NODES

Our next set of experiments investigated the robustness of our method to attack through poisoning of data at local nodes in the federated system. By scheduling training through the use of meta-gradients, we hypothesised that there would be an extra layer of redundancy which would prevent immediate poisoning of the model. In this experimental setup we split our dataset randomly (the size of the splits is also random). We only keep one node clean, with the rest of the datasets on the other nodes

being replaced with random values. Figure 6 shows learned scheduling for a student being trained on the MNIST dataset. We see how the scheduler begins with selecting a corrupted dataset before quickly transitioning to selecting another corrupted dataset. The scheduler then selects a dataset with clean data and this selection dominates for the rest of training. In Figure 7 we see that due to the initial training on corrupted data, the test-set performance degrades. However, as soon as the scheduler learns to use the clean data, the performance improves rapidly. It can also be noticed that the performance achieved is below state-of-the-art. This is likely due to there being a much smaller diversity in trainin data due to node corruption.

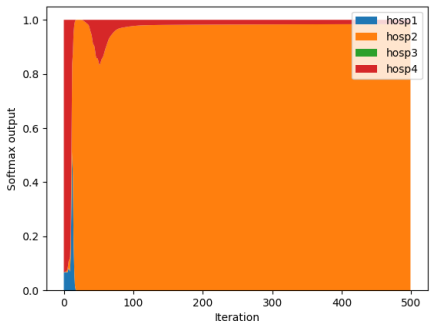

Figure 6: The scheduler selection for the different nodes in the federated system. We see that the scheduler learns to select the only clean dataset for training. The magnitudes of the different colours indicate the softmax output of the scheduler.

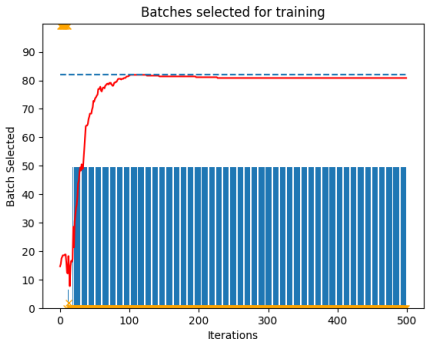

Figure 7: The teacher data selection as well as the performance of the student on the held-out test set for the MNIST digit recognition problem. The orange ‘x’ and blue bars represent the first and second outputs of the teacher respectively. The dashed line shows the maximum accuracy achieved. The red line is the performance of the student on the held-out test set.