1. Examples of Policies Learned

Learned Curricula

Here we present some examples of the curricula that were learned by the teacher for the three datasets we have used. We show that the policies learned are consistent according to the dataset and reflect a strategy that has been learned by the teacher.

WARD ADMISSION

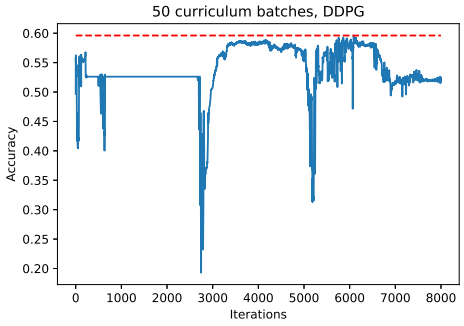

Figure 1. The performance of the student on the held-out test of the ward admission dataset while it is trained by the teacher. The red dashed line is the best performance achieved by this student.

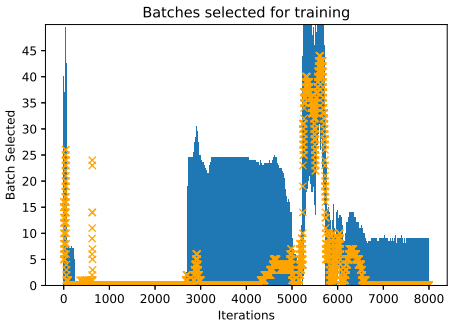

Figure 2. The actions generated by the policy of the teacher that has led to the performance of the student shown in Figure 1. Orange crosses are the first output (where to select data from) and blue bars are the second output (how much data around the central selection point to include in the batch for training). If the batch selected is near zero then this is low entropy data and if it is near the top of the batch selection then this is high entropy data.

We show another example of training by spiking in entropy to escape local minima in Figures 1 and 2. Once again there is a spike in entropy of data selected for training prior to 6000 iterations, which allows us to escape a local minimum and degrade the performance but upon further training achieve a better accuracy on the held-out test set. It would seem that this entropy spiking strategy is the preferred strategy for the ward admission dataset.

MIMIC-III

Plotted below are various examples of the curricula that were developed to train students on the MIMIC-III prediction problem. All of these provided state-of-the-art performance on the prediction problem.

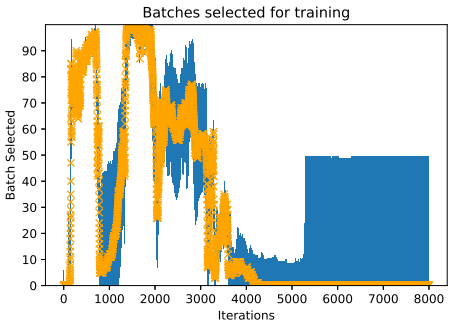

Figure 3. Curriculum generated for a randomly initialised student trained on the MIMIC-III dataset.

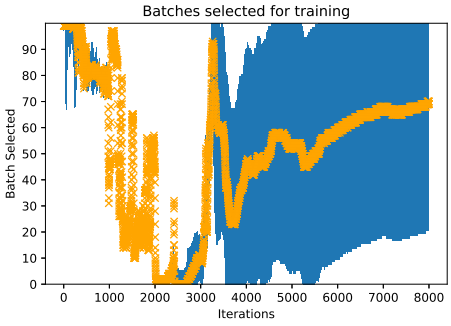

Figure 4. Curriculum generated for a randomly initialised student trained on the MIMIC-III dataset.

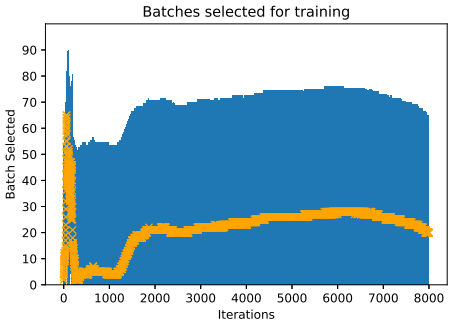

Figure 5. Curriculum generated for a randomly initialised student trained on the MIMIC-III dataset.

In Figures 3 and 4 we see that the teacher utilises very small data batches to train. This generally gives rise to very noisy training gradients which it seems the teacher uses to

converge to a favourable 'initialisation' from which it then starts to train on bigger batches. In Figure 5 we see that the teacher seems to bring the student into a 'good initialisation' early and so the rest of training is on the bigger batches.

CIFAR-10

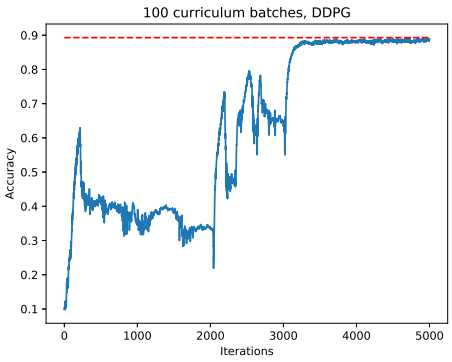

Figure 6. The performance of the student on the held-out test of the CIFAR-10 dataset while it is trained by the teacher. The red dashed line is the best performance achieved by this student.

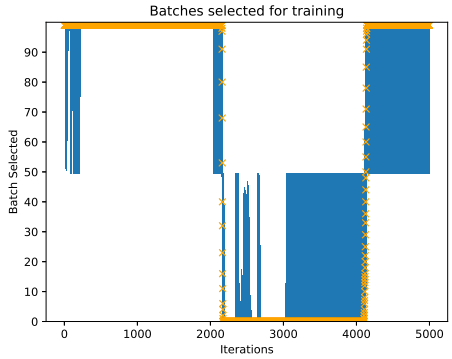

Figure 7. The actions generated by the policy of the teacher that has led to the performance of the student shown in Figure 6.

The performance of a student and the curriculum learned for training this student on the CIFAR-10 dataset are shown in Figures 6 and 7. We see the teacher primes the student into an initial state before (at approximately iteration 3000) repeatedly presenting low entropy batches before progressing to high entropy batches. This is very similar to curricula that are commonly used in many studies on image recognition. Figures 8 and 9 show the curricula used for other students by the same teacher. It would seem that repeated presentation of low entropy batches before progressing to repeatedly presenting high entropy batches is most beneficial for training the image recognition students. This makes sense due to the need for feature extraction in order to generalise to other images.

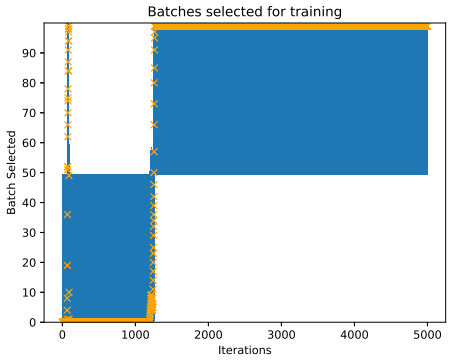

Figure 8. Curriculum generated for a randomly initialised student trained on the CIFAR-10 dataset.

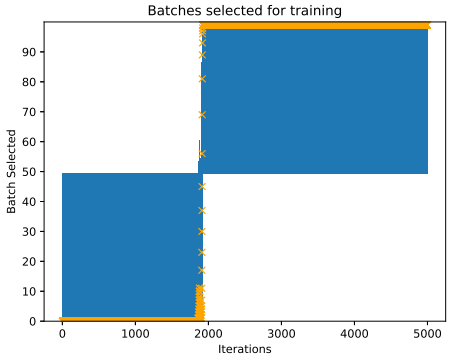

Figure 9. Curriculum generated for a randomly initialised student trained on the CIFAR-10 dataset. 1.

Constrained Policy Learning

In this section we present our findings of the policies of the teacher networks on various students for different tasks. We present the findings on the CIFAR-10 dataset in the main paper and the findings on the MIMIC-III and Ward Admission datasets below.

MIMIC-III

To constrain our students we first constrain our teacher (as done in the main paper) to select a batch width of zero with probability 0.999. Figure 10 shows the policy of the teacher when training the student on MIMIC-III data. When comparing these to typical MIMIC-III generated curricula (Figures 3, 4, 5), we see that there is no oscillation in entropy at the early stages of training and instead the teacher has learned to simply gradually step down in entropy. The student is trained with a learning rate of 0.02 and so in order to constrain this further we also reduce the students learning rate to 0.002, now constraining the student. We see from Figure 11 that the teacher begins training the student

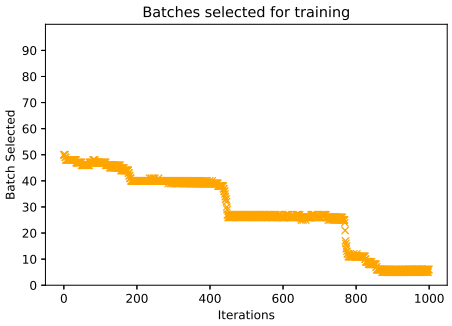

Figure 10. The actions generated by the learned policy of a constrained teacher to train a student on the MIMIC-III dataset. The student has a learning rate of 0.02.

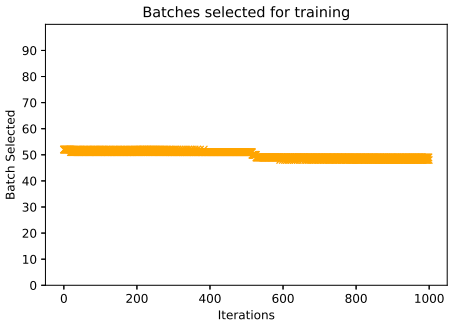

Figure 11. The actions generated by the policy of a constrained teacher to train a student that is also constrained with a lower learning rate of 0.002. The student has the same initial seed as that trained using the policy shown in Figure 10.

using similar data (at approximately batch 50 on the entropy scale), however due to the student's lower learning rate the downward stepping takes significantly longer. This is highly encouraging as it shows that the teacher is following the same strategy as used in Figure 10 albeit over a longer number of iterations as we would expect.

WARD ADMISSION

In Figures 12 and 13 we utilise a DQN trained teacher on the Ward Admission dataset. We initially train normally and then slow the learning rate of the student by 100 times for the same initial seed to see how this alters training.

We can see in Figure 12 that a 'recurring low to high entropy' curriculum is implemented by the teacher as seen implemented by the DDPG teacher. These can be seen as the DQN equivalent of the high entropy spiking strategy found by the DDPG teacher. Where we see drops in the entropy of data being used seem to be locations where the teacher is attempting to escape local minima. In Figure 13

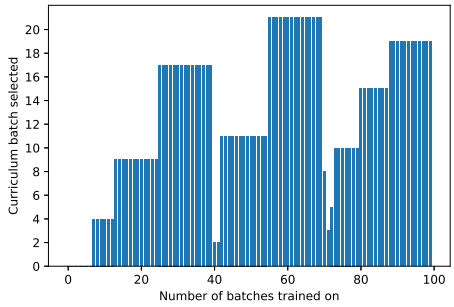

Figure 12. Actions generated by a DQN teacher with a learning rate of 0.01. At each iteration, anything shaded in blue is included in the batch used for training.

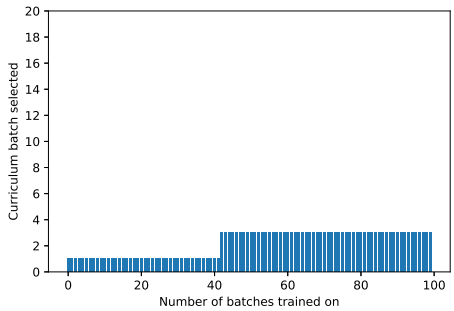

Figure 13. Actions generated by a DQN teacher with a learning rate of 0.001.

we reduce the student's learning rate and we see that we still have a 'low to high entropy' curriculum but it is progressing much more slowly. Once again, this is due to the step size being smaller and therefore requiring more gradient updates to get the student network into a weight state that requires different batches for training.

Policy Stability for Similar Students

We demonstrate in this section that for all the tasks considered our teacher learned stable policies conditioned on the current state of the student. We present our findings on the Ward Admission dataset in the main paper and our findings on the MIMIC-III and CIFAR-10 datasets below.

MIMIC-III

We see that once again the teacher learns a policy of using low entropy data to initialise the student before increasing the size of the batch introduced to maximise performance. We now once again apply Gaussian noise to the states of the student as done in the main paper.

In Figure 15 we see that the overall structure of the curriculum is the same as other MIMIC-III policies generated,

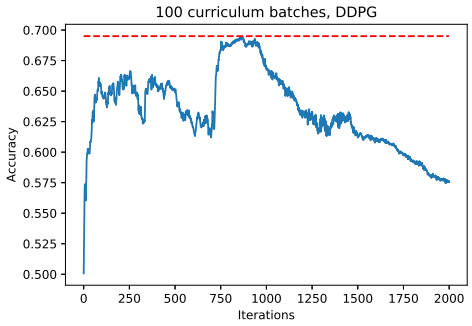

Figure 14. The performance of a student trained by the DDPG teacher on the MIMIC-III dataset.

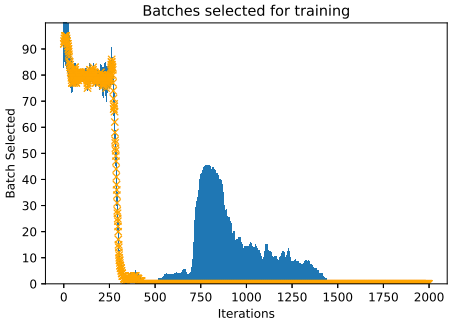

Figure 15. The actions used by the teacher to train the student with performance shown in Figure 14.

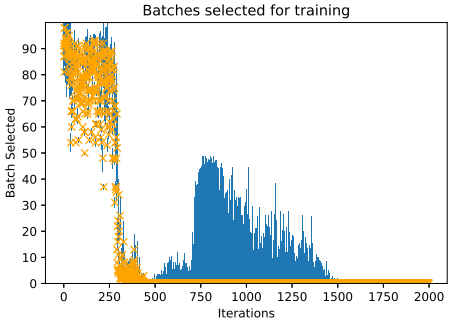

Figure 16. The actions taken by the teacher when the student has Gaussian noise applied to its states.

beginning at high entropy and reducing to low to initialise the student before expanding the size of the batch. Figure 16 also shows this with a very similar curriculum to the one in Figure 15 being followed. This further encourages us that a strategy has indeed been learned by the teacher to train a student on the MIMIC-III dataset based on the weights of the student.

CIFAR-10

Once again we repeat the exercise on the CIFAR-10 dataset and observe the stability of the teaching policy based on the corrupted states of the student.

Figure 17. The actions used by the DDPG teacher to train a student on the CIFAR-10.

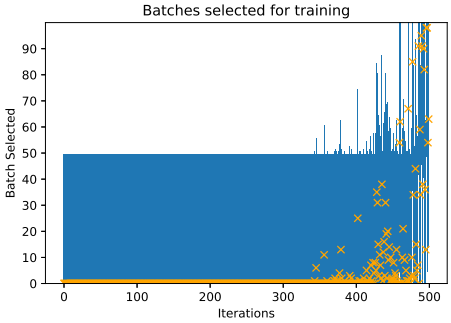

Figure 18. The actions of the teacher when the student has Gaussian noise applied to its states.

We see in Figures 17 and 18 that the same general policy is followed as that used in Figures 7, 8 and 9. As we only train for 500 iterations the policy ends at the point of transition to training on high entropy data. We see that corrupting the students states with Gaussian noise has not significantly changed the policy of the teacher, providing further reassurance that the policy is not only stable but a learned function of the state of the student and not simply an alternative optimisation trajectory.

Policy Transfer between Tasks

In this section we provide further examples of policies generated from a teacher trained using the Ward Admission dataset on the MIMIC-III mortality prediction task.

Figures 19 and 21 show the performances of two randomly initialised students and the corresponding curricula that generated these performances are found in Figures 20 and 22 respectively. We see from these policies that the teacher uses the same strategy of small batches for initialisation

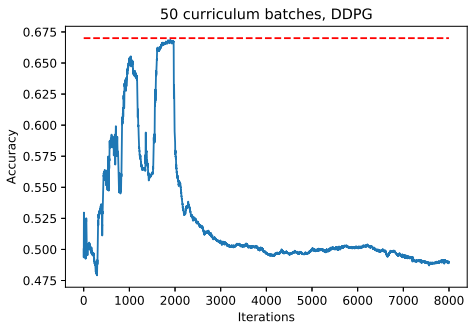

Figure 19. Performance of a randomly initialised student on the MIMIC-III dataset when trained by a teacher transferred from the Ward Admission dataset.

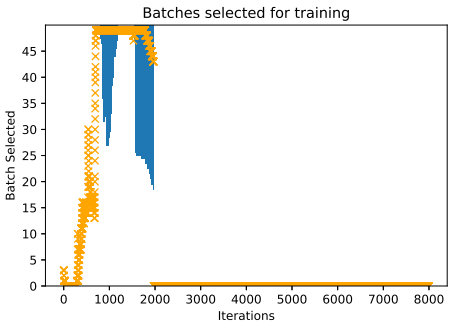

Figure 20. The actions selected by the transferred teacher when training the student for the MIMIC-III task.

and then the 'batch expansion' as was seen when discussing the policies of the teacher trained using MIMIC-III. It is interesting to see that the teacher for Ward Admission also demonstrates this behaviour, however it is not clear why this is the case. In future, we will investigate how to combine teachers to train tasks that may be combinations of tasks or hybrid tasks and assess the curricula generated from these. We may also make the problem hierarchical, with a principal assigning teachers or combinations of teachers to train various students on tasks which can be ranked according to some metric (such as a task embedding). This metric can then be related back to the specialties of the teachers, with the principal using this information to use multiple teachers (one iteration at a time) or combinations of teachers to train the student on the task.

Convergence of Teacher Selection

In this experiment we investigate how the teacher makes selections given a particular state of the student. In Figure 23 we monitor the output of the teacher for a given student state as the teacher is trained.

We show the teacher selection for 10 different initial seeds of

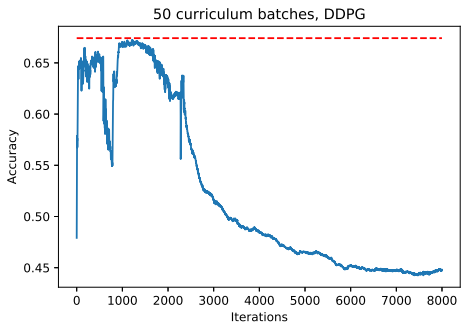

Figure 21. Performance of a randomly initialised student on the MIMIC-III dataset when trained by a teacher transferred from the Ward Admission dataset.

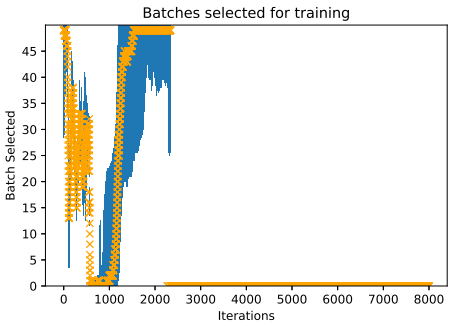

Figure 22. The actions selected by the transferred teacher when training the student for the MIMIC-III task.

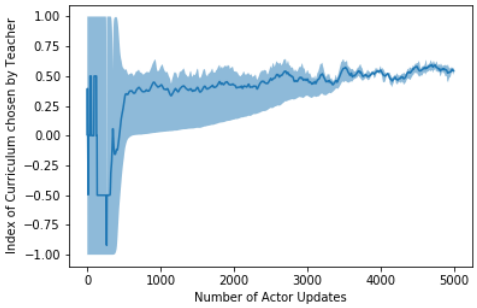

Figure 23. Action selection of the teacher on a fixed student state as it trains.

the teacher. We see that with training all teachers converge on a very specific action for the given student state indicating that a robust and consistent policy is learned.

2. Pseudocode for DQN Teacher

Algorithm 1 The student-teacher training routine for discrete batches using the DQN algorithm

Data: Training dataset organised into N batches of Mahalanobis curriculum
initialise teacher network, g
initialise target teacher by copying predictor teacher, g^T
select value of frequency of target network update and batchsize of replay data, M
for x *in* X *students* **do**
  initialise student network, f_x
  for i *in* I *iterations* **do**
    Extract state of f_x, s
    if i = *0* **then**
      train student on random batch (action), a
    else
      select a with highest Q-value from $g(s)$ according to a linearly decaying ϵ-greedy policy with respect to I
    end
    •train student (f_x) on action selected
    •record performance improvement of student on training set and validation set and multiply for overall reward, r
    •add r to the output of $g^T(s)$ corresponding to the action taken to achieve this reward
    •use the error between outputs of g and gT to backpropagate over the weights of g
    •save s, a, r and next state, s' into replay buffer
    if i *mod* M = *0* **then**
      sample M samples from replay buffer to train g on
      update g^T with new state of g
    else
      continue
    end
  end
end