Knowledge distillation for mixed domain classification

Ankur Agarwal

August 20, 2021

1 Introduction

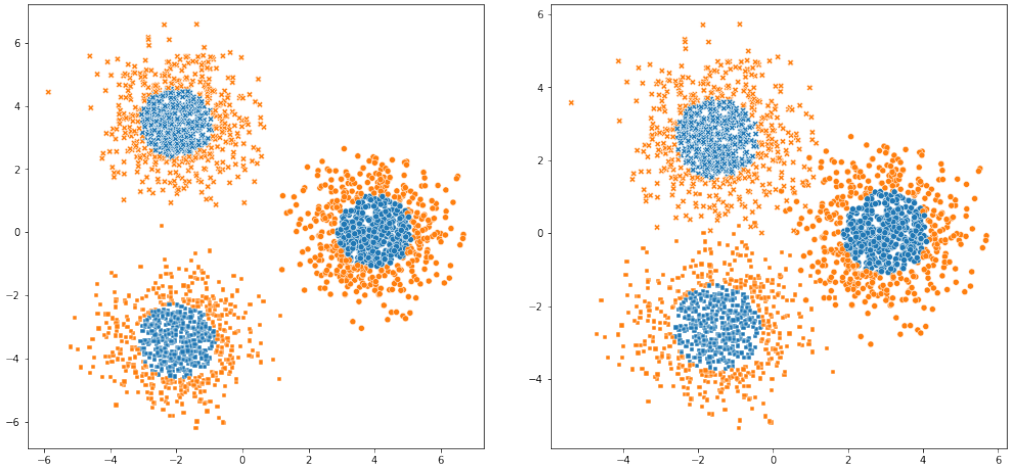
The objective of the assignment is to understand the benefit of knowledge distillation in a setting of teaching a student model to learn tasks of multiple domains by the help of expert teacher models for individual domains. The advantage of the technique is also learnt to be helpful when the multi-domain gets complicated and complex.

The objective is achieved by breaking down the assignment into 3 tasks for the complete understanding. At first, a single model is attempted to be trained in a mixture of all the domains. This helps in getting a baseline performance capability of the model. For second, teacher models are trained to specialize in each domain after which the distillation training is performed using the hard and soft labels. Further for a bonus round, additional tricks and techniques are allowed for a degree of freedom to improve the performance. Ideally, the improvement in the performance should follow an ascending trend to justify the benefit of expert teacher distillation training.

The metric of evaluation is accuracy. The validation accuracy is used as a measure of performance. The experiments are run using 5 seed values to make it reproducible and to have a statistical performance measure of the model.

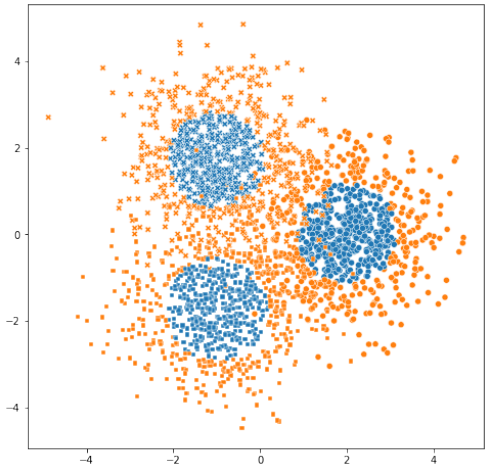
2 Data

The assignment is worked on a binary classification data of 3 different domains. The input data points have 2 feature columns defined by the 2D coordinates of their placement. There are 4 different variations of the data based on how close the centroid of the point cloud clusters are for the different domains. There are 2400 training data points and 300 validation data points equally balanced in terms of classes and domains.

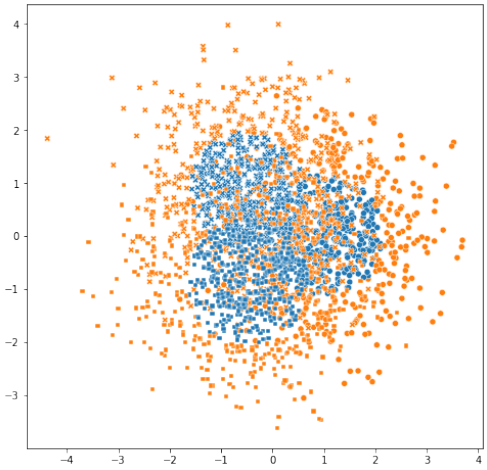


(a) Maximum separation

(b) 75% separation of (a)



(c) 50% separation of (a)



(d) 25% separation of (a)

Figure 1: Data plots of the 4 variations

3 Implementation Details

3.1 Experimental setup

The model used for any of the experiments is a 2 layered fully-connected Neural Network (FFCC) having hidden layers of 20 and 10 units. ReLU activation functions are used. The final layer is of 2 units with softmax functions for classification. Adam optimizer with a learning rate of 1e-3 is used as asked for the assignment.

The experiments are repeated for 5 given seed values [0, 10, 99, 1234, 2021] seeding the weight initialization of the FFCC network. For statistically stating the model performance based on the 5 seeded experiments, the mean and standard deviation of the validation accuracy is reported.

The training has been followed with batch gradient descent, i.e. the entire training set is passed as a single batch which would allow better and faster convergence to global minima. This is done considering randomization (since the data points are ordered as batches of individual domains) would require a seed for reproducible while also the fact that batch gradient descent would have better gradient estimates than mini-batch or stochastic gradient descent.

```python
class FFNN(nn.Module):
    """The desired 2 hidden layer feed forward neural network."""
    def __init__(self, random_state=0):
        super(FFNN, self).__init__()

        # Seeding the random weight initialization of the network.
        seeder(random_state)

        # Layer Initialization
        self.hidden1 = torch.nn.Linear(2, 20)
        self.hidden2 = torch.nn.Linear(20, 10)
        self.out = torch.nn.Linear(10,2)

    def Z(self, x):
        """
        Returns:
            Logit values on forward pass which need to be passed to the softmax for classification.
        """
        z = self.hidden1(x)
        z = torch.relu(z)
        z = self.hidden2(z)
        z = torch.relu(z)
        return self.out(z)

    def forward(self, x):
        """
        Returns:
            Log-softmax values on forward pass to the network.
        """
        logit = self.Z(x)
        return torch.log_softmax(logit, dim=1)
```

Figure 2: Model definition and seeding operation.

3.2 Methodology

For the first task, a regular cross entropy loss is used as the classification loss.

$$\text{Categorical CE}(y, S(X)) = L_{CE} = -\sum_{i=1}^{n=2} y_i \log(S(X)_i)$$

$$\implies \text{Binary CE}(y, S(X)) = L_{CE} = -(y \log(S(X)_i) + (1 - y_i) \log(1 - S(X)_i))$$

For the second task of teacher-student learning we use the cross entropy loss for hard labels and the KL divergence loss for soft labels,

$$L_{KD} = D_{KL}[S(Z(X)_t/\tau)||S(Z(X)_s/\tau)]$$

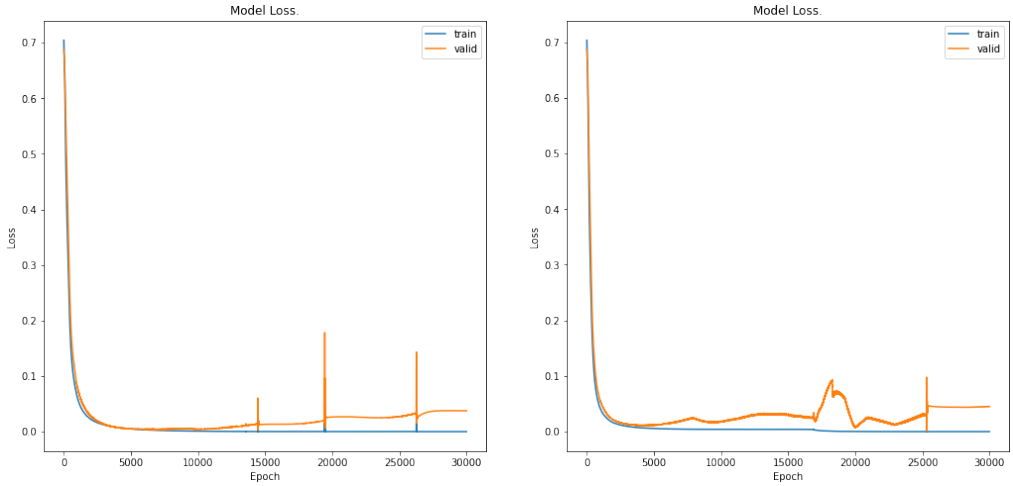
In order to improve gradient flow, the divided temperature term while taking the derivative is multiplied again to cancel it out. Similarly, log softmax is used instead of softmax for the training student model.

$$L_{KD} = D_{KL}[\log(S(Z(X)_t)/\tau)||S(Z(X)_s/\tau)] \cdot (\tau^2)$$

$$L = \alpha \cdot L_{KD} + (1 - \alpha) \cdot L_{CE}$$

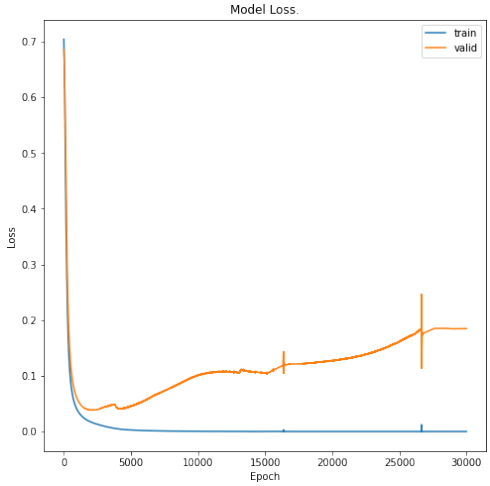
For improving the results as for the 3rd task, two methods are provided.

Method 1: The alpha hyper parameter is tuned to bring minor improvements. The domain expert teacher is found to be overfitting ad the validation loss diverges by the final epoch hence not providing with the most generalised teacher model. The same is the case for student model as well. To improve model selection, a method as simple as early stopping is implemented to pick up the checkpoint with the least validation loss hence providing with the most generalized teacher and student model. This adds on to the improvements.

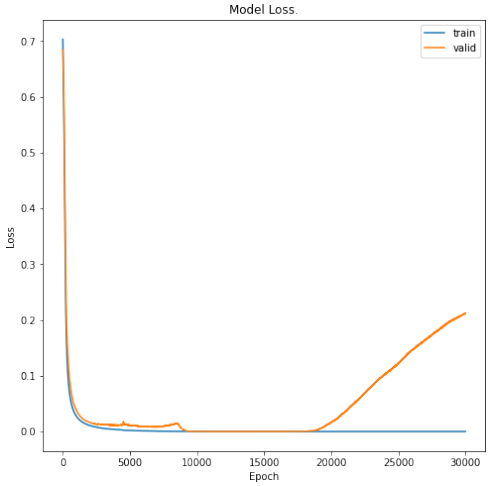


(a) Maximum separation data

(b) 75% separation of (a) data



(c) 50% separation of (a) data



(d) 25% separation of (a) data

Figure 3: Teacher loss curves on domain 0 training shows consistent overfitting pattern. Random seed 0. See appendix for other domain plots.

Method 2: As the data gets more complicated with the mixing of the different domains, learning from perfect teachers become equally difficult as achieving the functional complexity of the teacher model is completely spending its capacity for one domain and the same is not possible with a single student model so easily in a domain mixture. The loss curve would have a lot of sharp curvatures and many local minima. In order to smoothen the descent, the method of slowly learning by learning the more commonly strong features first is used by scheduling a decreasing decreasing temperature value.

The temperature value has a max value when the loss curvature is expected to be sufficiently smooth that increasing the temperature value makes not much difference on the learning curve. The temperature value is reduced until 1 after which the training is again performed for fine tuning using alpha weighting of hard and soft labels. The approach is motivated from the knowledge distillation annealing work. The early stopping method is used here as well.

$$\phi(epoch) = \begin{cases} \frac{1+\cos \pi \cdot (epoch/epochs)}{2} \cdot \tau_{max} & , \text{if } \frac{1+\cos \pi \cdot (epoch/epochs)}{2} > 1 \\ 1 & , \text{if } \frac{1+\cos \pi \cdot (epoch/epochs)}{2} \leq 1 \end{cases}$$

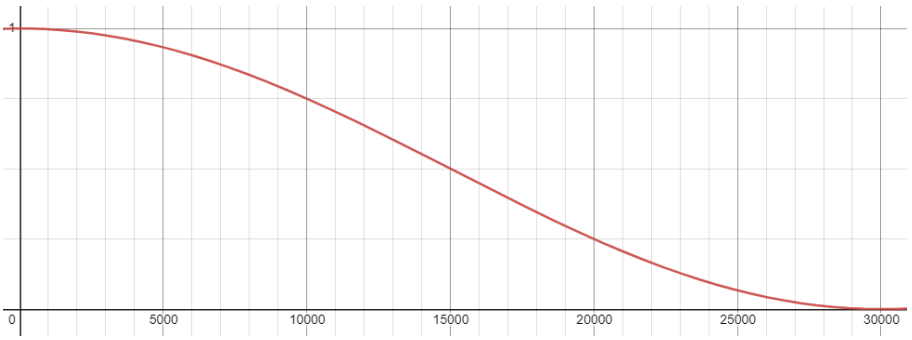


Figure 4: Cosine annealing function as a function of the epoch for controlling $\tau_{max} \geq \phi \geq 1$.

$$L_{KD\_MSE} = ||Z(X)_t - Z(X)_s \cdot \phi||_2^2$$

$$L = \begin{cases} L_{KD\_MSE} & , \text{if } \tau_{max} \geq \phi > 1 \\ \alpha \cdot L_{KD\_MSE} + (1 - \alpha) \cdot L_{CE} & , \text{if } \phi \leq 1 \end{cases}$$

4 Results

It was assumed that the seeds are to be used in common for the teacher as well as the student, i.e., the same weight initialization of the teacher and the students. The brought 5 possible combinations and the experiments were repeated for those. The following are the mean and standard deviations of the validation accuracy performances. Except for the dataset with 25% separation, question 3 has been solved for the rest using method 1 successfully. More hyper-parameter tuning was required to achieve success for method 2 which was not possible due to limitation of time and hardware resources.

Data Seperation	Q1	Q2	Q3 (Method 1)	Q3 (Method 2)
100%	99.53% +/- 0.001825	99.46% +/- 0.001825	99.80% +/- 0.001825	93.46 +/- 0.008315
75%	99.40% +/- 0.002788	99.60 +/- 0.002788	99.66% +/- 0.002357	93.73 +/- 0.048327
50%	97.8% +/- 0.001825	97.99% +/- 0.003333	98.13% +/- 0.001825	85.40% +/- 0.058089
25%	78.80% +/- 0.013662	77.33% +/- 0.013333	78.06% +/- 0.007226	67.39 +/- 0.032778

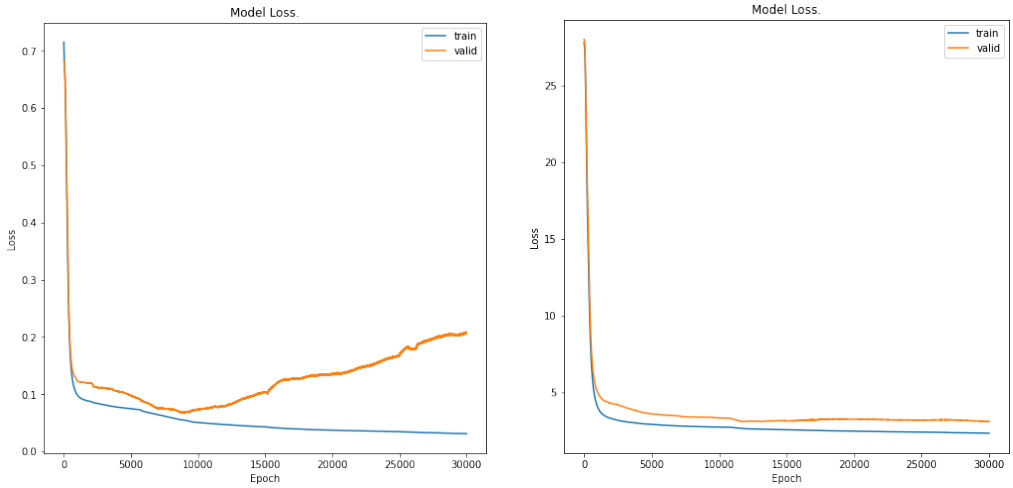
Table 1: The following are the tabulated results of the mean and standard deviation of the validation accuracy for the 5 seeded runs for different datasets and questions.

5 Discussion

From the results we can see that the distillation method of teaching a student model from domain expert teacher models (question 2 & 3) outperforms the same student trained using only hard labels without the teacher models (question 1). The exception holds for the data with 25% of the separation due to its complexity where the outperforming question 1 results occur from a mere luck of the model being able to find a hack using its capacity focused to only fit the training data in any way. This can be justified looking at the variance error in the student model's training for question 1 (Table 6). This variance error can be seen to reduce in question 2 & 3 relatively (Figure 5, 8, 9, 10).

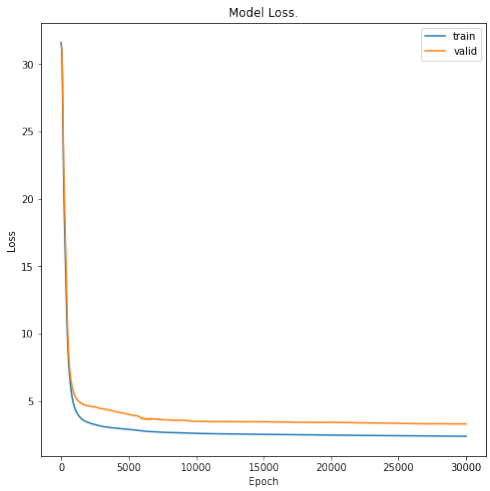
The reason for the question 2 & 3 results to be better in general is because of the use of soft label training. The distillation technique from the logits of domain expert models allows in depth learning for the student model to achieve the implicit features of the expert model. Further softening using temperature allows loosening the constraints to such features and hence smoothens the learning curve. This smoothing of the curve is enough to allow the student to learn from multiple domain experts. Additionally, the features learnt in this distillation setting compared to a student solely learning a mixture domain classification allows more generalization as the logits represent implict features of expert models which are able to see the individual domains independent of the noise of multiple domains.

The trend seen while tuning the temperature hyper-parameter is that the loss curve becomes flatter and flatter to a certain limit as and when the temperature is increased. The generalization trend noticed for the same is a decreasing variance error and then an increasing variance error. The variance and bias trade off is also consistently followed here. The harder the data complexity of mixed domains gets, the higher becomes the need for the temperature hyper-parameter to be. The alpha parameter is used to shift weights between the hard and soft labels as per the need of the model to see the complete picture of the mixture domains learnt from the hard labels compared to individual domain view learnt from the soft labels.

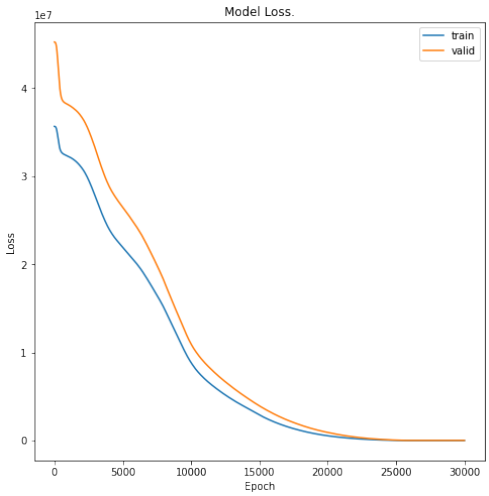


(a) Question 1

(b) Question 2



(c) Question 3 (Method 1)



(d) Question 3 (Method 2)

Figure 5: Student loss curves showing variance error on multi domain training over different question for data with 50% separation. Random seed 0. See appendix for other data variation student loss curve plots.

6 Conclusion

To conclude, the expert teacher based distillation learning allows the student model to generalize better by learning important rather than just strong features and hence improving the overall performance of the model to classify in a multiple domain setting. The annealing technique of varying the temperature is makes sense as it reinforces a scheduled procedure for feature learning from multiple domains. All this adds on to the benefit of guided feature learning attained from distillation learning.

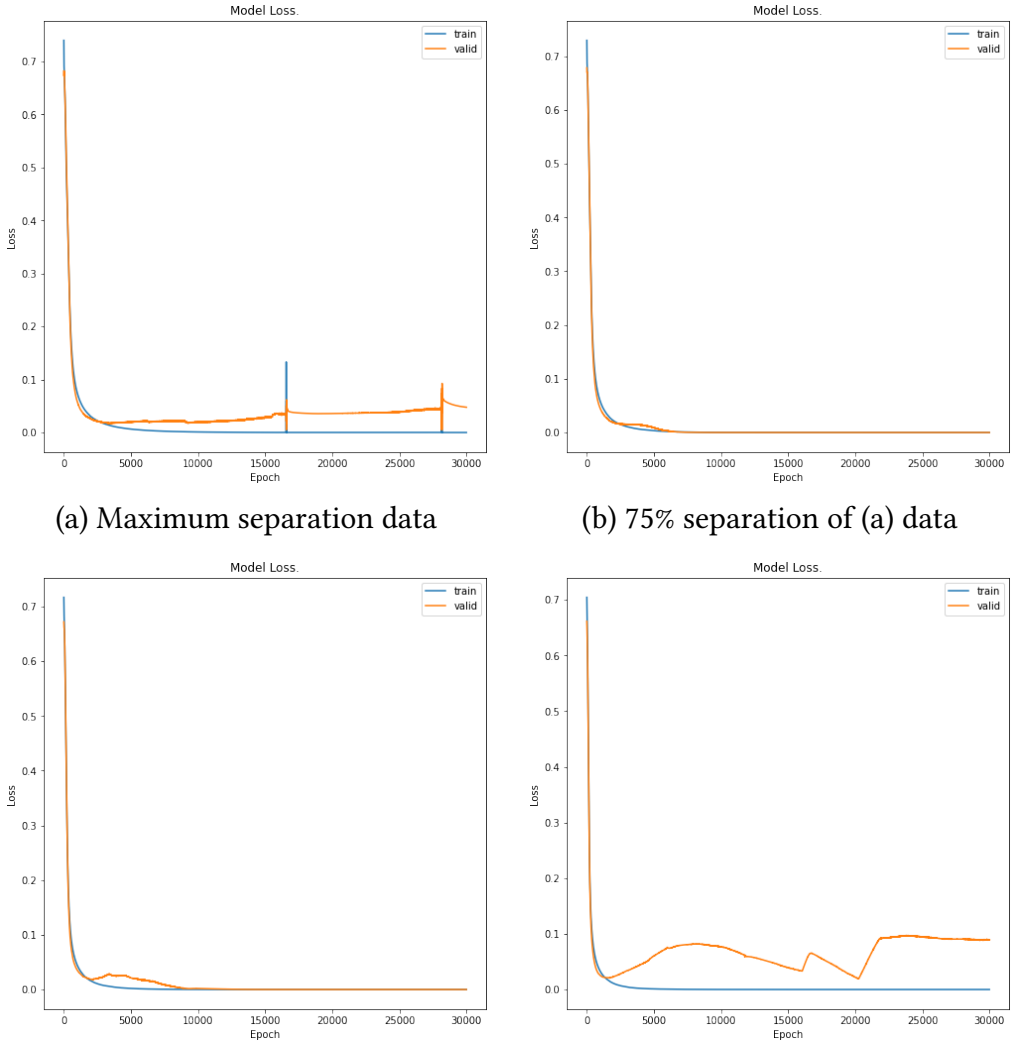
For future works of improvement a few additional things that could be done are adding batch normalization to the FFCC before the ReLU activations, using AdamW with AMSgrad and the 1 cycle policy learning scheduler for super convergence.

For reproducibility, the code and jupyter notebook is provided with all the experimental setups in the github repository: *https://github.com/ankur-98/Multi-domain-Knowledge-Distillation*

7 References

1. Hinton, G., Vinyals, O. and Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.
2. Jafari, A., Rezagholizadeh, M., Sharma, P. and Ghodsi, A., 2021. Annealing Knowledge Distillation. arXiv preprint arXiv:2104.07163.
3. AdamW and Super-convergence is now the fastest way to train neural nets by Sylvain Gugger and Jeremy Howard.

8 Appendix



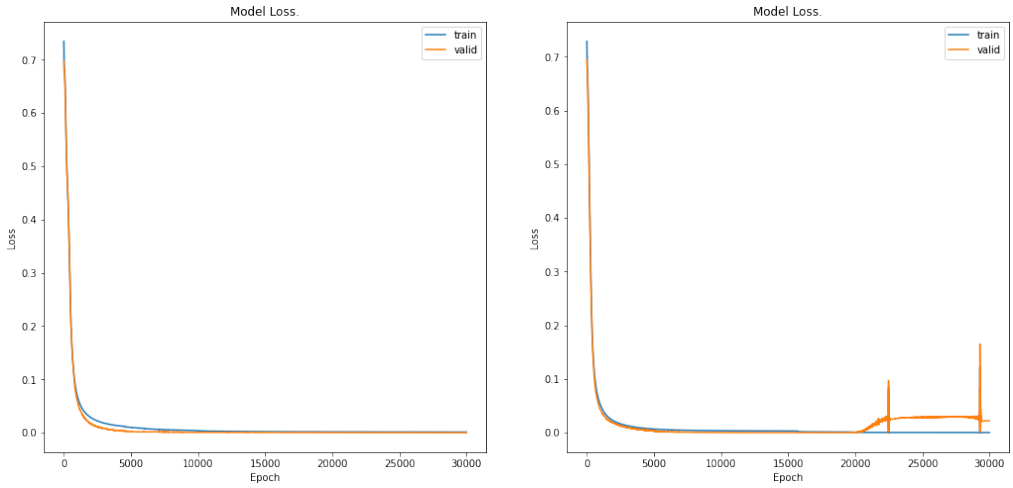
(a) Maximum separation data

(b) 75% separation of (a) data

(c) 50% separation of (a) data

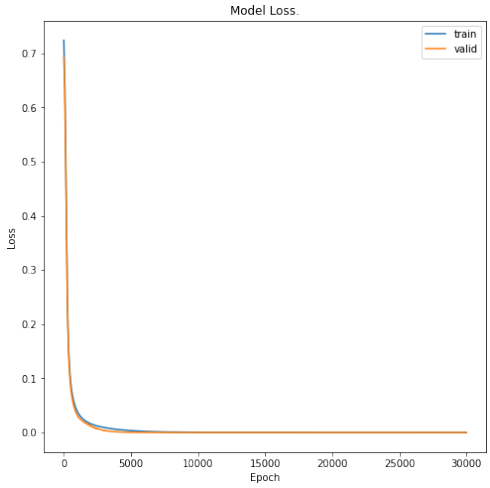
(d) 25% separation of (a) data

Figure 6: Teacher loss curves on domain 1 training. Random seed 0.

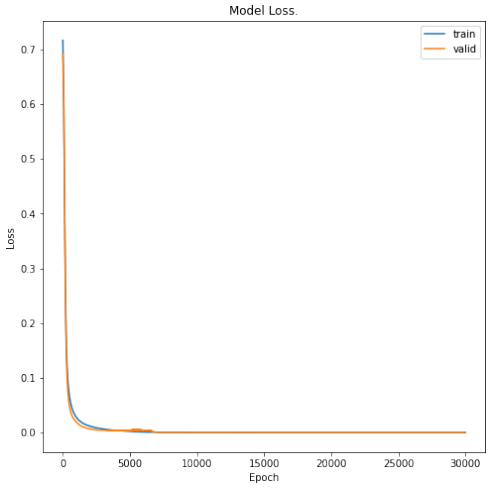


(a) Maximum separation data

(b) 75% separation of (a) data



(c) 50% separation of (a) data

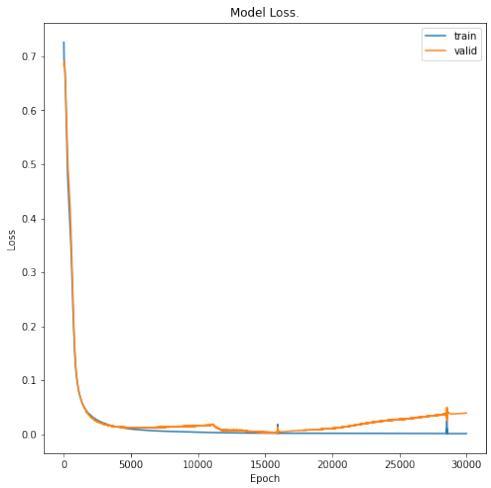


(d) 25% separation of (a) data

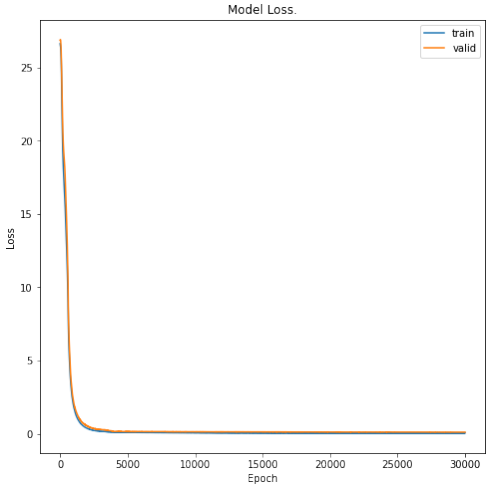
Figure 7: Teacher loss curves on domain 2 training. Random seed 0..

Data Seperation	**Teacher Domain 0**		**Teacher Domain 1**		**Teacher Domain 2**	
	Train	**Valid**	**Train**	**Valid**	**Train**	**Valid**
100%	99.87%	100%	100%	100%	100%	100%
75%	99.87%	100%	100%	100%	100%	100%
50%	99.62%	98%	100%	100%	100%	100%
25%	100%	100%	100%	98.99%	100%	100%

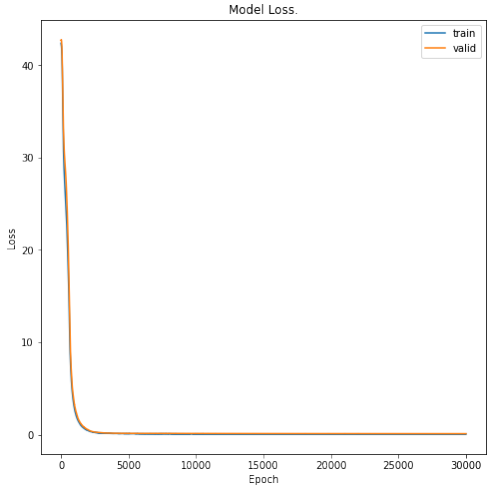
Table 2: Accuracy of teacher expert models on training with early stopping. Random seed 0.



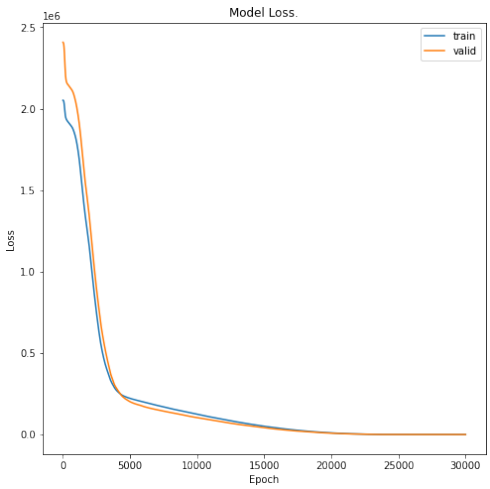
(a) Question 1



(b) Question 2

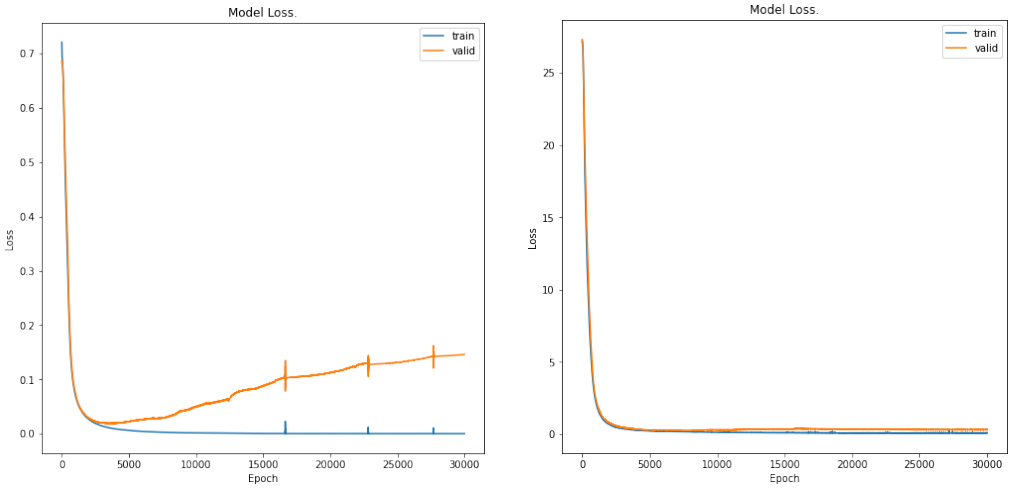


(c) Question 3 (Method 1)



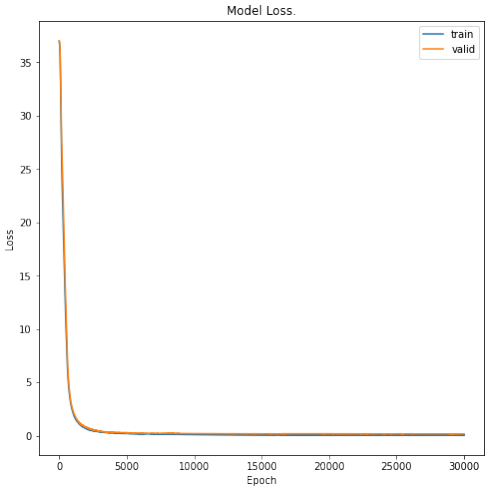
(d) Question 3 (Method 2)

Figure 8: Student loss curves showing variance error on multi domain training over different question for data with 100% separation. Random seed 0. See appendix for other data variation student loss curve plots.

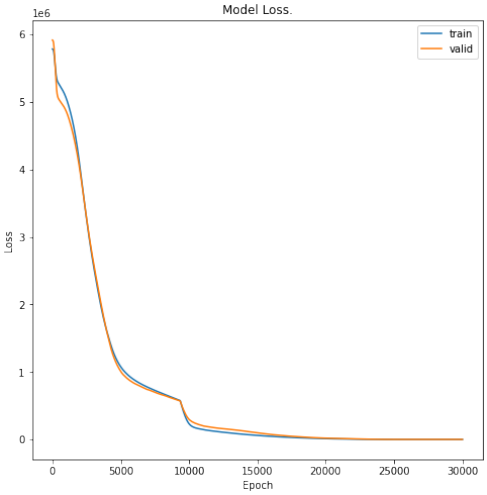


(a) Question 1

(b) Question 2

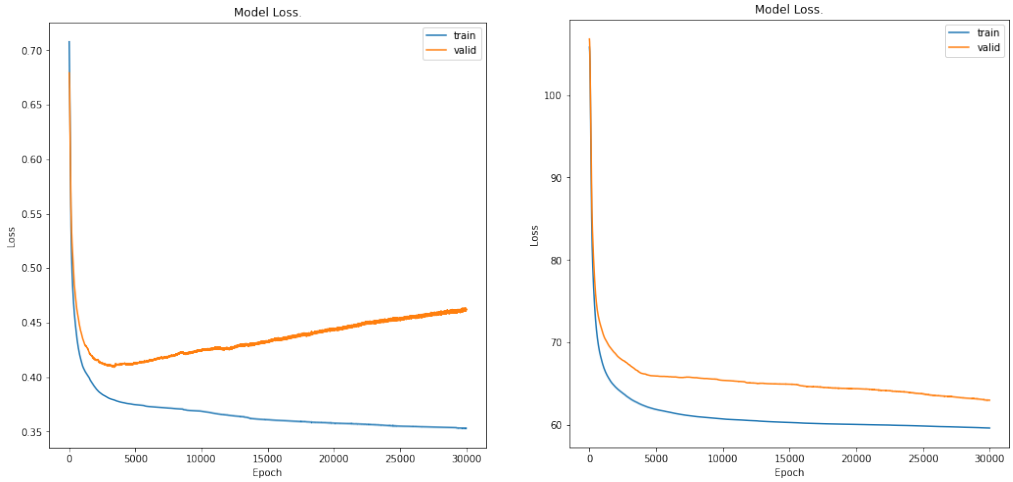


(c) Question 3 (Method 1)



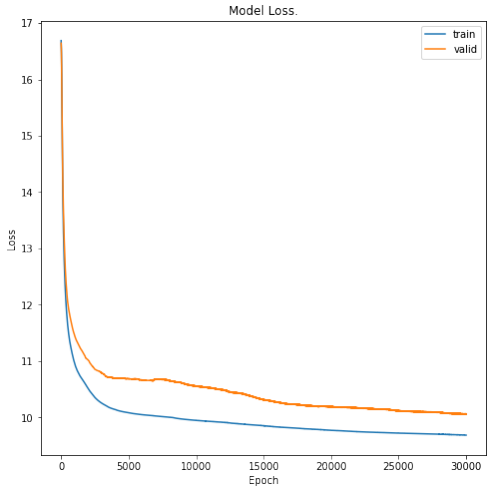
(d) Question 3 (Method 2)

Figure 9: Student loss curves showing variance error on multi domain training over different question for data with 75% separation. Random seed 0. See appendix for other data variation student loss curve plots.

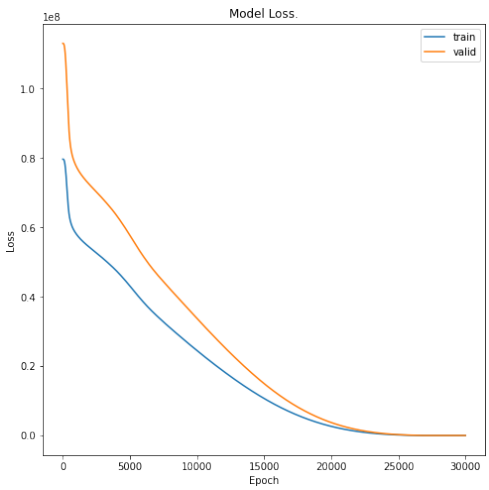


(a) Question 1

(b) Question 2



(c) Question 3 (Method 1)



(d) Question 3 (Method 2)

Figure 10: Student loss curves showing variance error on multi domain training over different question for data with 25% separation. Random seed 0. See appendix for other data variation student loss curve plots.

Random Seed	Q1		Q2		Q3 (Method 1)		Q3 (Method 2)	
	Train	Valid	Train	Valid	Train	Valid	Train	Valid
0	99.95%	99.66%	99.95%	99.33%	99.91%	99.66%	99.20%	99.33%
10	100%	99.33%	99.87%	99.33%	99.54%	100%	98.37%	97.66%
1234	100%	99.66%	99.95%	99.66%	99.25%	100%	96%	97.33%
99	99.95%	99.33%	99.74%	99.66%	99.33%	99.66%	82.45%	79.00%
2021	100%	99.66%	99.91%	99.33%	99.33%	99.66%	94.70%	94%

Table 3: Accuracy of student models on 100% separation data.

Random Seed	Q1		Q2		Q3 (Method 1)		Q3 (Method 2)	
	Train	Valid	Train	Valid	Train	Valid	Train	Valid
0	100%	99.33%	99.62%	99.33%	99.83%	99.66%	97.75%	97.66%
10	100%	99.66%	99.95%	99.66%	99.37%	99.66%	81.74%	86.33%
1234	100%	99.33%	99.95%	100%	99.20%	99.66%	92.20%	91.33%
99	99.91%	99.66%	100%	99.66%	99.70%	99.33%	97.79%	97%
2021	100%	99%	100%	99.33%	99.66%	100%	95.45%	96.33%

Table 4: Accuracy of student models on 75% separation data.

Random Seed	Q1		Q2		Q3 (Method 1)		Q3 (Method 2)	
	Train	Valid	Train	Valid	Train	Valid	Train	Valid
0	99%	98%	98.29%	98.33%	98.16%	98%	90.04%	90.33%
10	98.66%	97.66%	98.37%	97.66%	98.29%	98%	91.62%	92.66%
1234	98.79%	97.66%	98.45%	98%	97.66%	98%	79.41%	80.33%
99	98.33%	98%	98.75%	98.33%	98.04%	98.33%	86.20%	83.66%
2021	98.75%	97.66%	98.29%	97.66%	98.04%	98.33%	82.16%	80%

Table 5: Accuracy of student models on 50% separation data.

Random Seed	Q1		Q2		Q3 (Method 1)		Q3 (Method 2)	
	Train	Valid	Train	Valid	Train	Valid	Train	Valid
0	83.20%	79%	81.20%	78%	81.70%	78.66%	68.08%	69.66%
10	82.12%	78.33%	80.33%	75.33%	81.66%	77.33%	61.70%	62.66%
1234	83.20%	78.33%	81.25%	78%	81.29%	79%	60.25%	65.33%
99	82.79%	77.33%	80.87%	76.66%	80.99%	77.66%	70.62%	70.33%
2021	82.70%	81.00%	81.74%	78.66%	81.125%	77.66%	66.62%	69%

Table 6: Accuracy of student models on 25% separation data.