Q-Match: Self-Supervised Learning by Matching Distributions Induced by a Queue

Thomas Mulc TMULC@GOOGLE.COM
Google

Debidatta Dwibedi DEBIDATTA@GOOGLE.COM
Google Deepmind

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

In semi-supervised learning, student-teacher distribution matching has been successful in improving performance of models using unlabeled data in conjunction with few labeled samples. In this paper, we aim to replicate that success in the self-supervised setup where we do not have access to any labeled data during pre-training. We introduce our algorithm, Q-Match, and show it is possible to induce the student-teacher distributions *without* any knowledge of downstream classes by using a queue of embeddings of samples from the unlabeled dataset. We focus our study on tabular datasets and show that Q-Match outperforms previous self-supervised learning techniques when measuring downstream classification performance. Furthermore, we show that our method is sample efficient–in terms of both the labels required for downstream training and the amount of unlabeled data required for pre-training–and scales well to the sizes of both the labeled and unlabeled data.

Keywords: Self-Supervised Learning

1. Introduction

Tabular data is the most common form of data for problems in industry. While many robust techniques exist to solve real-world machine learning problems on tabular data, most of these techniques require access to labels. Leveraging unlabeled data to learn good representations remains a key open problem in the tabular domain. In this work, we propose a flexible and powerful framework using deep learning that helps us use unlabeled data in the tabular domain.

Deep learning has been successful in processing data in many different domains like images, audio, and text. Learning non-linear features using deep architectures has been shown to be a key feature in improving performance across a wide variety of problems like image recognition (Krizhevsky et al., 2017), speech recognition (Deng et al., 2013), and machine translation (Singh et al., 2017). Until recently, achieving state-of-the-art performance would not have been possible without large, manually annotated datasets. However, a class of learning algorithms called self-supervised learning (Wu et al., 2018; Devlin et al., 2018) has shown that highly performant features can be learned without large-labeled datasets as well. In this work, we propose a new self-supervised learning algorithm and study its effectiveness in the tabular data domain.

In self-supervised learning, a task known as the *pretext* task is first solved on a dataset that is typically large and unlabeled. The aim of self-supervised learning is to learn an encoding function f parameterized by θ that captures variances and invariances in the dataset without the need for human-annotated labels. This initial training is usually referred to as the pre-training stage. The parameters learned from the pretext task during pre-training are then used to solve a new objective called the *downstream* task. Concretely, f is a function with parameters θ that maps a sample x from the input dimensionality d' to an embedding size d that is $f : (X, \theta) \mapsto R^d$ where x is a single data-point in the input space X. After the pretext algorithm updates the model parameters θ during the pre-training stage, the downstream data (typically a manually annotated dataset) is either used to fine-tune θ or learn the parameters of a linear classifier on the output of f.

Recently proposed self-supervised approaches for tabular data (Darabi et al., 2021; Yoon et al., 2020; Arik and Pfister, 2021; Lee et al., 2020) have shown encouraging results. In this work, we propose a novel method for self-supervised learning called Q-Match which is closely related to the semi-supervised learning framework called FixMatch (Sohn et al., 2020). For labeled data, FixMatch uses the standard supervised learning loss. For unlabeled data, FixMatch proposes to match the student and teacher distributions over the set of classes used in the downstream task. In a self-supervised setup, however, access to the relevant classes or any labeled data during pre-training is limited. For this reason, Q-Match uses a queue of embeddings instead of known classes to induce the teacher and the student distributions. The queue is implemented similar to MoCo He et al. (2020) and NNCLR (Dwibedi et al., 2021) by updating a list of embeddings with newer embeddings and discarding older ones as training proceeds. We use individual samples (via their embeddings) to generate the student and teacher distributions required for training. We find Q-Match leads to improvements not only in the final performance of the downstream model, but also reduces the number of labeled samples required for the downstream task.

2. Proposed Approach

Motivation. Our self-supervised approach is based on the success of the semi-supervised learning algorithm FixMatch (Sohn et al., 2020). In their method, the authors show it is possible to leverage a large unlabeled dataset and a few labeled samples to improve the performance of the model on a downstream task. They do so by matching the class distributions of the student and teacher *views* produced by augmenting the input in two different ways. We hypothesize that it might be possible to adapt their framework to the self-supervised setup by removing the dependency on the known classes during training. To do so, we keep a queue of past embeddings that can serve as a proxy for *classes*. We use this queue to produce the target and student distributions used to train a model. The training then proceeds by performing continuous knowledge distillation from the teacher to the student model such that the student ultimately learns to predict the distribution induced by the teacher.

Method. In Figure 1, we outline our method for the pretext training approach used in Q-Match. We corrupt the input x_i two times independently using the method proposed in VIME (Yoon et al., 2020) to produce the student view $x_{i,s}$ and the teacher view $x_{i,t}$. We pass $x_{i,s}$ through the student model to produce the student embedding $z_{i,s}$. Similarly, we pass $x_{i,t}$ through the teacher model to produce the teacher embedding $z_{i,t}$. We want this pair of embeddings to induce similar probability distributions over a representative set of samples of the dataset. In other words, we want our encoder to maintain similar relationships between

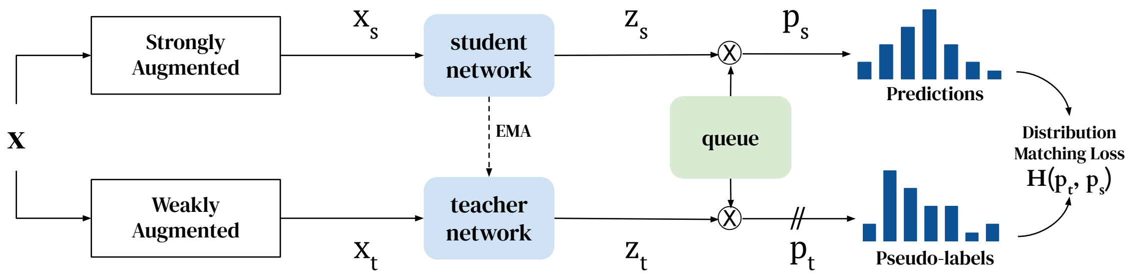


Figure 1: **Q-Match Training Method.** The network is trained by performing continuous self-distillation by minimizing the cross-entropy between student-teach distributions. The two slanted lines denote a stop-gradient operation. The queue is updated with z_t at each training step.

different data samples in spite of the corruption introduced while generating the views. As the dataset can be quite large, we maintain a fixed size queue Q of past embeddings like MoCo (He et al., 2020) and NNCLR (Dwibedi et al., 2021). A stop-gradient is applied to the teacher embeddings, so only the student weights are updated at each iteration. We multiply the student embeddings and the teacher embeddings with the embeddings in the queue to produce the student and teacher logits. After taking the softmax of these logits, we produce the student distribution, $p_{i,s}$, and the teacher distribution, $p_{i,t}$. The teacher distribution, $p_{i,t}$, is the target distribution which the student distribution, $p_{i,s}$, should match. We define the distribution matching loss as

$$\mathcal{L}_i^{\text{QM}} = H(p_{i,t}, p_{i,s}) = H\left(\text{softmax}\left(\frac{z_{i,t} \cdot Q}{\tau_t}\right), \text{softmax}\left(\frac{z_{i,s} \cdot Q}{\tau_s}\right)\right)$$

where τ_t is a scalar temperature value that controls the sharpness of the distribution produced by the teacher, τ_s is a scalar temperature value that controls the sharpness of the student distribution, $Q \in \mathbb{R}^{d \times m}$ is the queue of m many previous teacher embeddings, and $H(p, q)$ is the cross-entropy loss between two distributions p and q. We normalize the views $z_{i,s}$ and $z_{i,t}$ using L2 normalization. The queue size is constant and is refreshed at every training iteration with the previous batch of teacher embeddings while the oldest embeddings are removed from the queue. The parameters of the teacher model are updated using the Exponential Moving Average (EMA) of the student model parameters. Empirically, we observe that corrupting the teacher with a smaller probability leads to better performance (discussed later in Section 3.3).

Model Architecture. In all experiments, we use an MLP as our encoding function f. We follow the same architecture from i-Mix (Lee et al., 2020) which includes five fully-connected layers (2048-2048-4096-4096-8192). The final layer uses a 4-set max-out activation (Goodfellow et al., 2013). All layers except the output layer have batch normalization followed by a ReLu. We use a linear projector of 128 dimensions on top of the encoder in all our experiments. Similar to i-Mix, we added a 2-layer (512-128) MLP projection head, but we noticed that it performed similarly to the model without the projection head for Q-Match training.

Data Preprocessing. For data preprocessing, we compute the normalization statistics by using a batch normalization layer without the learnable parameters of scale and bias just after the input layer. During evaluation, the accumulated exponential moving average statistics are applied to the original inputs normalizing them. We found this method works just as well as normalizing the data by calculating the statistics on the full dataset. For categorical features, we one-hot encoded all categorical features. Additionally, we experimented with quantile transformation (Gorishniy et al., 2022) of the inputs. However, this only showed benefits in the Adult dataset. Hence, we do not use quantile transformation for other datasets.

3. Experiments

Datasets. In this work we use the following datasets: Higgs (Baldi et al., 2014), Cover Type Asuncion and Newman (2007), Adult (Kohavi et al., 1996) and MNIST (LeCun et al., 2010). We take various subsets of the original datasets for different experiments. For the exact splits used in different experiments, please refer to Table 8 in the Appendix.
Implementation. We implement our method in JAX (Bradbury et al., 2018). The experiments were conducted on single V100 and P100 GPUs.

3.1. Comparison with Baselines

In this section, we want to measure how our method compares with other self-supervised methods that have been evaluated on tabular data. We find that different papers evaluate their approach on different datasets with different split settings. The pretext set size and labeled set size vary in the original experiments conducted in these papers. To be fair in our comparison to prior work, we report the performance of Q-Match with the same downstream dataset sizes as used by the original authors.
Higgs Dataset. First, we compare Q-Match against three other self-supervised methods for tabular data on the Higgs (Baldi et al., 2014) dataset: TabNet (Arik and Pfister, 2021), i-Mix (Lee et al., 2020), and CORE (Han and Ranganath, 2021). We report the results of this experiments in Table 1. Q-Match outperforms all the baselines under all the different splits. In particular, we observe that Q-Match outperforms TabNet with a pretext dataset size that is 100 times smaller.
Cover Type Dataset. Next, we compare the performance of our method with baselines on the Cover Type dataset (Asuncion and Newman, 2007). We report the results of our experiment in Table 2. There are two commonly used splits: Cover Type 10% and Cover Type 15k. In the fine-tuning setup, we observe that Q-Match outperforms Contrastive MixUp by a margin of about 10%. We hypothesize this gain in performance is due to the fact that in the contrastive loss the encoder mistakenly considers samples belonging to the same class as negatives. The Q-Match loss does not suffer from this problem. The encoder is free to learn the similarities between different samples and does not consider all items in the batch as negatives. In the linear evaluation setup, we observe Q-Match is about 0.7% to 1.6% better than two versions of i-Mix that use the contrastive loss. Q-Match is slightly worse (0.3%) than i-Mix BYOL in the linear evaluation setup.
MNIST 10% Dataset. Next, we perform a similar comparison on the MNIST (LeCun et al., 2010) dataset. While it is an image dataset, prior work (Yoon et al. (2020); Darabi et al. (2021) has used MNIST 10% for research in tabular domain by converting the pixels in an image into a flattened vector. In the past, only 10% of the training set is used as labeled data while the rest 90% of the training set is used as pretext data. We report the results of this experiment in Table 3. We observe that while Q-Match outperforms VIME by a margin

Method	**Pretext Examples**	**Labeled Examples**	**Accuracy**
CORE (Han and Ranganath, 2021)	50k	5k	66.92 ± 0.55
Q-Match (Ours)	50k	5k	**68.13** ± **1.02**
TabNet (Arik and Pfister, 2021)	10M	10k	68.96 ± 0.39
Q-Match (Ours)	10M	10k	**71.13** ± **0.21**
TabNet (Arik and Pfister, 2021)	10M	100k	73.19 ± 0.15
i-Mix (Lee et al., 2020)	100k	100k	72.9
Q-Match (Ours)	100k	100k	**73.27** ± **0.19**

Table 1: **Higgs experiments.** CORE, TabNet, and Q-Match all report fine tuning accuracy, while i-Mix reports the linear classification accuracy.

of about 2%, our method is comparable in performance with the Contrastive Mixup method without using the MixUp augmentation.

Method	**Cover Type 10% Accuracy**	**Cover Type 15k Accuracy**
Constrastive MixUp (Darabi et al., 2021)	80.41 ± .205	-
i-Mix N-Pair (Lee et al., 2020)	-	72.1 ± 0.2
i-Mix MoCo v2 (Lee et al., 2020)	-	73.1 ± 0.1
i-Mix BYOL (Lee et al., 2020)	-	**74.1** ± **0.2**
Q-Match Finetune (Ours)	**90.26** ± **.11**	82.76 ± .07
Q-Match Linear (Ours)	-	73.79 ± .22

Table 2: **Cover Type experiments.** For Cover Type 15k, i-Mix uses a linear classifier on top of the pretext features. For Constrastive MixUp, the evaluation uses fine tuning. We present results of Q-Match for both tasks.

Method	**Accuracy**
VIME (Yoon et al., 2020)	95.77 ± .22
Constrastive MixUp (Darabi et al., 2021)	97.58 ± .08
Q-Match (Ours)	**97.67** ± **.21**

Table 3: **MNIST 10% experiments.** Fine tuning accuracy for VIME, Constrastive MixUp, and Q-Match.

3.2. Data Scaling Experiments

In the previous subsection, we show that Q-Match either outperforms or is at par with the other self-supervised baselines when the pretext size is large and all the samples in the labeled downstream dataset are used for evaluation. In this experiment, we want to measure how these methods perform if the amount of labeled data and the pretext dataset size changes.

To compare fairly against all the baselines, for this set of experiments, we re-implement the following methods: TabNet, VIME, and i-Mix (N-Pair version).

Training Details. In all the experiments in this section, we first train the encoder on the pretext task. We then proceed to the downstream tasks which can either be fine-tuning or linear evaluation. We always compare against the supervised learning baseline. This baseline refers to the method where we do not train a model with any pretext task but begin downstream task training from random initialization. We perform a grid search over relevant hyper parameters for each method. For all methods, we search over pretext task and downstream task learning rates. For the TabNet, VIME, SimCLR, SimSiam, DINO, VICReg, and Q-Match algorithms, we added the probability of corrupting a column (as defined in Yoon et al. (2020)) to the grid. For the N-Pair i-Mix algorithm, we also add the loss temperature to the grid. For the Q-Match algorithm, we also add the queue size and the student temperature to the grid. Please refer to Table 7 in the Appendix for more details on the parameters. We pick the best parameters according to the validation dataset for each downstream task. For all experiments, we report the average and standard deviation over 5 trials. We train all pretext tasks using a maximum of 200 epochs with early stopping and a patience of 32 (using the pretext validation dataset). For the supervised tasks, we train for a maximum of 500 epochs with a patience of 32 on the validation accuracy. All tasks use a batch size of 512 and the parameters are updated using Adam citepadamw optimizer with weight decay. All algorithms used zero weight decay during pretraining and 10^{-1} weight decay during the downstream tasks. We make our code[1] publicly available for further details.

Corruption Function. One important factor that affects the performance is the choice of the corruption function used to create the two augmented views. Both VIME and TabNet augment the orignal data by randomly corrupting columns. The VIME corruption function replaces corrupted values with samples from the pretext dataset while the TabNet corruption function replaces values with zeros. We always use the VIME corruption function in all our experiments. Even for our implementation of the TabNet algorithm, we use the VIME corruption function as we found the VIME corruption performed better. Unless otherwise stated, we do not corrupt the teacher view and only corrupt the student view.

Few Shot Learning. In this experiment, we compare the representations learned by different learning algorithms by only using 1% of the total available labels for evaluation. This scenario is common in industry when number of available labels for a downstream task might be less but there might a lot of unlabeled data available to learn an encoder. In addition to the tabular self-supervised algorithms we used as comparisons for other experiments, we also include results from our implementations of the following methods that have previously been used for image self-supervision tasks: SimCLR (Chen et al., 2020b), SimSiam (Chen and He, 2021), DINO (Caron et al., 2021), and VICReg (Bardes et al., 2021). For SimCLR, we also include a large batchsize (4096) version, since this has been found to be helpful (Chen et al., 2020b). For all datasets, we only use one-percent of the original labels for downstream training and keep the rest of the data for pretext training and validation. We report the performance of learning a linear classifier in Table 4 and fine-tuning the entire encoder in Table 5. For the Linear Classification Task, Q-Match outperforms all other methods on all datasets except VICReg on MNIST, which it performed similarly. For the Finetuning task, Q-Match was competitive on all datasets, but did especially well on the CoverType and Higgs datasets.

1. https://github.com/google-research/google-research/tree/master/q_match

	Dataset				Rank
Algorithm	Cover Type 1%	Higgs100k 1%	Adult 1%	MNIST 1%	
Supervised Baseline	70.45 ± 0.18	60.39 ± 0.39	78.63 ± 0.20	85.97 ± 0.15	5.3
TabNet	51.28 ± 3.51	61.15 ± 0.56	76.46 ± 0.41	24.20± 6.57	8.8
DINO	57.18 ± 4.61	56.87 ± 1.66	76.84 ± 0.85	64.63 ± 3.59	8.5
i-Mix	67.90 ± 0.45	60.19 ± 0.40	75.62 ± 0.79	90.66 ± 0.32	7.3
SimCLR (large batch)	66.87 ± 0.25	64.22 ± 1.08	76.66 ± 1.95	91.01 ± 0.68	6.0
SimSiam	64.66 ± 1.22	60.11 ± 1.55	78.75 ± 2.28	92.98 ± 0.53	5.8
VIME	68.42 ± 0.31	64.37 ± 0.62	79.01 ± 2.26	88.02 ± 0.59	4.3
VICReg	64.86 ± 0.16	65.81 ± 0.34	76.67 ± 1.93	**97.36 ± 0.25**	4.3
SimCLR	69.66 ± 0.16	65.42 ± 0.22	76.87 ± 0.52	91.84 ± 0.16	3.8
Q-Match (Ours)	**70.90 ± 0.36**	**66.84 ± 0.34**	**80.33 ± 0.47**	97.13 ± 0.23	**1.3**

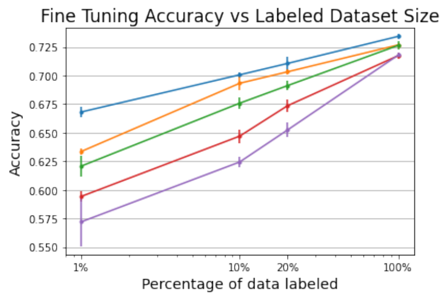
Table 4: Linear classification accuracy of our method versus other pretext algorithms.

	Dataset				Rank
Algorithm	Cover Type 1%	Higgs100k 1%	Adult 1%	MNIST 1%	
Supervised Baseline	71.63 ± 0.25	58.41 ± 0.14	78.14 ± 0.72	88.98 ± 0.30	6.8
TabNet	70.58 ± 0.63	61.80 ± 1.00	77.25 ± 1.28	86.70 ± 1.33	8.0
SimCLR (large batch)	69.73 ± 0.71	58.18 ± 1.99	77.72 ± 1.66	92.54 ± 0.62	8.0
SimSiam	71.80 ± 0.11	60.87 ± 1.96	68.99 ± 6.52	93.05 ± 0.46	6.0
i-Mix	71.30 ± 0.35	61.98 ± 0.45	77.83 ± 0.88	92.00 ± 0.22	6.0
DINO	**72.43 ± 0.5**	55.76 ± 3.17	78.22 ± 0.66	89.58 ± 0.93	5.8
VICReg	68.66 ± 0.34	60.86 ± 4.09	**80.36 ± 0.29**	**97.47 ± 0.10**	5.0
SimCLR	71.77 ± 0.30	63.49 ± 0.60	79.09 ± 0.78	93.13 ± 0.27	3.5
VIME	71.51 ± 0.24	64.42 ± 0.80	80.50 ± 1.74	93.54 ± 0.12	**3.0**
Q-Match (Ours)	72.42 ± 0.37	**66.03 ± 1.01**	77.75 ± 1.89	96.44 ± 0.41	**3.0**

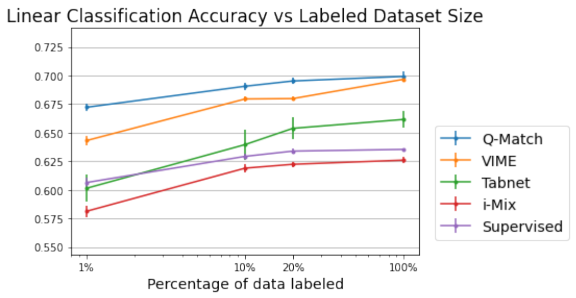
Table 5: Finetuning accuracy of our method versus other pretext algorithms.

Varying Downstream Dataset Size. In this experiment we want to measure how downstream task performance changes as more labels are available. We increase the fraction of labeled data available in the Higgs dataset and train the downstream task. Note that in this experiment the pretext size is fixed for all methods at 100k samples. We report the results of this experiment in Figure 2. We observe that Q-Match outperforms other methods across different fractions of labeled data in both the fine-tuning and linear evaluation tasks. The difference in performance between Q-Match and other methods increases as the number of labeled samples becomes less. In other words, Q-Match is more sample efficient in terms of labels required. Also note that in the low labeled data regime, self-supervised pre-training using Q-Match provides a good initialization for the downstream task. This initialization leads to about 10% improvement in performance than simply performing supervised learning from random initialization.

Varying Pretext Dataset Size. In this experiment we want to measure how downstream task performance changes as more unlabeled data is available for the pretext task training. We increase the fraction of unlabeled data available in the Higgs dataset and run both the pretext training and downstream task training. Note that in this experiment the downstream labeled set size is fixed for all methods at 100k samples. We report the results of this experiment in Figure 3. We observe that Q-Match outperforms other methods. The difference in performance is especially stark when fewer samples are available. In other words, Q-Match is more sample efficient even in terms of unlabeled data requirements. Depending on the amount of unlabeled data available, Q-Match can increase the performance on downstream task by 5%-8% in the finetuning setup.

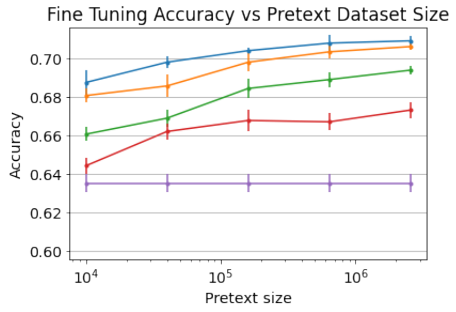


(a) Fine Tuning Classification Task

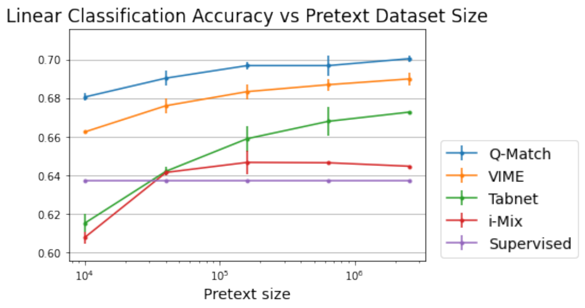


(b) Linear Classification Task

Figure 2: Classification performance as the size of the labeled data increases for the Higgs100k dataset.



(a) Fine Tuning Classification Accuracy



(b) Linear Classification Accuracy

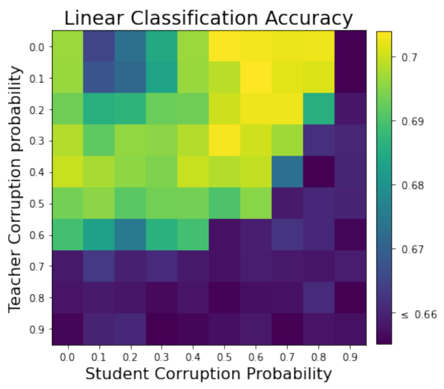
Figure 3: Accuracy for different sizes of the pretext set for the Higgs dataset.

3.3. Sensitivity Analysis

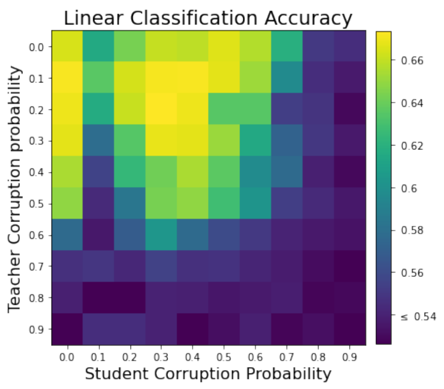
In this subsection, we study how some hyperparameter choices affect downstream task performance. In particular, we find our method's final performance is affected significantly by the corruption probability and the queue size. Below we report how sensitive the learning algorithm is for these datasets: Higgs and Cover Type.

Corruption Probability. We experiment varying the corruption probability in both views in Q-Match. The linear classification results are shown in Figure 4. The more yellow/bright the color of the grid, the better the performance while more blue/dark means the performance is worse. Note that in both the Cover Type and the Higgs experiments, there appears be both a maximum teacher and student corruption probability that is beneficial for pre-training with Q-Match. After this maximum value, there is too much corruption of the original input to learn a useful representation. Additionally, there is also a critical maximum value of the sum of both probabilities going beyond which, the model fails to learn useful representations using Q-Match. In other words if both the student and teacher are corrupted with high corruption probability, it leads to sub-optimal performance. *We recommend using a small or zero value for the teacher corruption, and a moderate value for the student corruption to achieve optimal downstream performance.*

On the other hand, we find the fine-tuning results to be fairly robust to these parameters. The initialization found by performing Q-Match with different values of corruption probability is good enough for downstream fine-tuning to achieve optimal performance.



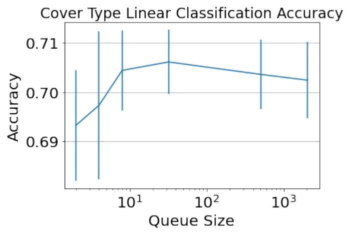
(a) Cover Type 1%



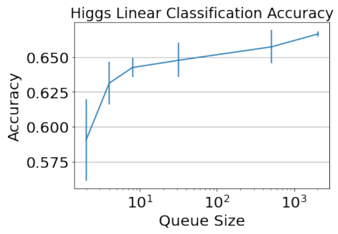
(b) Higgs 1%

Figure 4: Linear classification accuracy as corruption of the features changes for both the student and the teacher views.

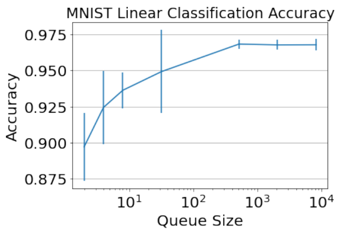
Queue size. We examine the effect of changing the size of the queue used in Q-Match for the Higgs, Cover Type, and MNIST datasets. The linear classification performance (in the few shot setting) as a function of the queue size is shown in Figure 5. First, we observe that when the queue size becomes very small, the final performance declines. Second, we note as the queue size increases, there is the less variance in the downstream results. *We recommend using a larger queue ($\geq 10^3$) to achieve optimal performance.*



(a) CovType 1%



(b) Higgs 1%



(c) MNIST 1%

Figure 5: Effect of the queue size on the linear classification accuracy.

3.4. ImageNet Experiments

As our proposed approach is domain-agnostic, it can also be used to learn image features in a self-supervised manner. We pre-train a ResNet-50 model with the Q-Match loss on the ImageNet dataset (Deng et al., 2009). We use the multi-crop training setup of NNCLR Dwibedi et al. (2021) and SWAV Caron et al. (2020) where 2 crops of 224×224 and 6 crops of 96×96 are used in a single batch. The two larger sized images serve as the teacher views. We pre-train the model for 800 epochs. We use a student temperature of 0.1 and teacher temperature of 0.04. We do not use any color jitter augmentation on the teachers' view to produce the weakly augmented view. Crop augmentation is used on both views. We use a queue size of 98304, embedding size of 256, momentum of 0.99, and the same projection MLP used in NNCLR.

We report the results of linear evaluation of the 2048-d output from the ResNet-50 model in Table 6. We find that training with the Q-Match loss results in better performance than training with the baseline methods. In particular, the performance improvement over DINO is interesting because both DINO and Q-Match use the student-teacher distribution matching loss. While DINO uses learnable prototypes to induce these distributions, Q-Match uses a queue of past embeddings. This further validates the utility of the queue in the student-teacher distribution matching framework. Furthermore, even though Q-Match was developed primarily with tabular datasets with a low number of downstream classes, it is capable of strong performance on image datasets with a larger number of classes.

Method	**Accuracy**
SWAV (Caron et al., 2020)	75.3
DINO (Caron et al., 2021)	75.3
NNCLR (Dwibedi et al., 2021)	75.6
Q-Match (Ours)	**76.0**

Table 6: **ImageNet Linear Evaluation.** Linear evaluation accuracy for NNCLR, SWAV, and Q-Match.

4. Discussion and Related Work

Contrastive Self-supervised Learning. In the context of self-supervised learning, a class of approaches that have been effective are built on top of the InfoNCE loss (Oord et al., 2018) where for any data point x_i and its corresponding embedding $z_i = f(x_i, \theta)$ there exists a set of positives P_i and negatives N_i, loss L_i^{InfoNCE} is defined as follows:

$$\mathcal{L}_i^{\text{InfoNCE}} = -\log \frac{\sum_{z^+ \in \mathcal{P}_i} \exp\left(z_i \cdot z^+/\tau\right)}{\sum_{z^+ \in \mathcal{P}_i} \exp\left(z_i \cdot z^+/\tau\right) + \sum_{z^- \in \mathcal{N}_i} \exp\left(z_i \cdot z^-/\tau\right)} \quad (1)$$

where $(z_i,\, z_i^+)$ are positive pairs, $(z_i,\, z^-)$ are negative pairs and τ is the softmax temperature. This loss aims to attract the positives closer to each other while repelling the negatives farther from each other. In SimCLR (Chen et al., 2020a), the positives are two views of the same data and negatives are all the other elements in the current mini-batch. In MoCo (He et al., 2020), the positives are the same as SimCLR but the negatives are derived from a queue of past embeddings produced by the model.

Both Contrastive Mixup and i-Mix N-Pair use the InfoNCE loss. We find Q-Match outperforms both these methods consistently. This could be due to the fact the contrastive loss mistakenly considers samples from the same class as negatives. For a dataset whose classes are uniformly distributed, the probability that at least one sample in the batch of negatives belongs to the same downstream class as the positives is equal to $1 - (\frac{N-1}{N})^{(B-1)}$ where N is the number of classes in the downstream task and B is the batch size used during training. This means the occurrence of this event is less in a large-scale and diverse image datasets like ImageNet. Concretely, with a batch size of 512 and the number of downstream classes equal to 1000, there is a probability of approximately 0.4 that an element of the same class will be considered a negative. However in the tabular setting where the number of classes in the downstream task is usually much less (say 10), the probability of considering items of the same class as negatives is much higher (very close to 1). We hypothesize that this might be the reason that using the vanilla N-pair contrastive loss usually results in sub-par performance.

Non-Contrastive Self-supervised Learning. In order to remove the dependency on explicit negatives during training, researchers have proposed two types of losses. The first class of approaches like BYOL(Grill et al., 2020), SimSiam (Chen and He, 2021) aim to directly minimize the distance between the positive embeddings using a Mean Squared Error (MSE) loss.

$$\mathcal{L}_i^{\text{MSE}} = -z_i \cdot z_i^+ \tag{2}$$

In BYOL (Grill et al., 2020), z^+ is derived from a different view that is passed through a momentum encoder while in SimSiam z^+ comes from a different view passed through the same encoder except it has a stop-grad operation to prevent embedding collapse. We compare Q-Match with i-Mix BYOL on the Cover Type dataset in Table 2 and find it is competitive with our method. But our implementation of i-Mix BYOL failed to achieve good performance. We leave exploring i-Mix BYOL on low labeled data regime as future work.

The second class of losses that do not require explicit negatives use prototypes to minimize the cross-entropy between two distributions induced by the positive pair. Examples of this approach are SwAV (Caron et al., 2020) and DINO (Caron et al., 2021).

$$\mathcal{L}_i^{\text{DINO}} = H(\frac{z_i^+ \cdot P}{\tau}, \frac{z_i \cdot P}{\tau}) \tag{3}$$

where P is a list of learnable prototypes maintained by the model. To avoid embedding collapse DINO uses momentum encoding with a combination of centering and sharpening while SwAV uses Sinkhorn clustering.

Like DINO and SWAV, Q-Match also uses the cross-entropy between two distributions to learn an encoder. But Q-Match differs from them in two aspects. First, while DINO and SWAV use learnable prototypes to induce the target distribution, we use a queue to produce the target distribution. Second, instead of relying on centering and sharpening or Sinkhorn clustering, we use a queue of past embeddings to prevent embedding collapse. Since the queue of embeddings keeps getting refreshed, it is non-trivial for the model to collapse. We believe this is the main reason Q-Match outperforms SWAV and DINO. It is potentially easier for the model to overfit to the prototypes to bring the self-supervised loss down but not as easy to do so with a queue that is continually refreshed.

Reconstruction-based Self-supervised Learning. A commonly used approach for self-supervised learning is using a reconstruction loss to solve a de-noising pretext task. Methods, like VIME (Yoon et al., 2020) and (Arik and Pfister, 2021) take the original sample in the data, and corrupt some of its values using a corruption mask sampled from a Bernoulli distribution. They then learn an encoding function as well as a reconstruction function which aims to reconstruct the original sample using both the learned parameters of the encoder and the parameters specific to the reconstruction task. TabNet is pre-trained to only predict the reconstructed values. On the other hand, VIME also proposes predicting the corruption mask in addition to predicting the reconstructed values. We find the corruption function introduced in these papers useful for producing student and teacher views in Q-Match. However, we find the distribution matching loss to be more effective in the low data regime than the reconstruction based losses.

Semi-supervised Learning Our approach is also closely related to the semi-supervised approaches like FixMatch (Sohn et al., 2020) and PAWS (Assran et al., 2021) that learn encoders by matching student-teacher distributions. The difference is that we focus on the self-supervised setup where the downstream task fine-tuning happens after the self-supervised pre-training stage. Hence, we cannot assume knowledge of any known classes during the pre-training stage. Instead, we use a queue to induce the student and teacher distributions. Additionally, we focus the problem in the context of tabular datasets while the above mentioned papers are evaluated on image datasets.

5. Conclusion

We introduced a new self-supervised algorithm, Q-Match, that learns new, useful representations of tabular data entirely from unlabeled data. Q-Match utilizes a queue to perform continuous self-distillation by matching the student distribution to the teacher distribution as training proceeds. We show that Q-Match outperforms existing self-supervised algorithms and supervised learning on tabular datasets in terms of classification accuracy. Additionally, we show that Q-Match is more efficient than existing methods in terms of sizes of the unlabeled pretext dataset and of the labeled downstream dataset. Q-Match continues to outperform other methods when the size of both the pretext and the downstream datasets is increased.

6. Broader Impact

Q-Match has minimal assumption about the distribution or the domain of data. Hence, it can be applied to data that is not tabular or a mix of tabular and non-tabular domains as well (see Appendix for ImageNet results). We show that Q-Match reduces the amount of unlabeled pretext data and labeled downstream data. This can be advantageous in domains when collecting data itself is expensive, when samples can only be labeled after waiting a long time, or even when the total number of possible labels are naturally limited (e.g., rare medical diagnoses).

On the other hand, even if the encoder is learned in a self-supervised manner without labels it might be biased against certain groups, especially if they are rarely represented in the data. It is recommended to visualize embeddings to identify such cases. Performing detailed group-wise analysis on the downstream task metric will also surface any bias learned by the encoder.

References

Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687, 2021.

Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8443–8452, 2021.

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1), jul 2014. doi: 10.1038/ncomms5308. URL https://doi.org/10.1038%2Fncomms5308.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

Sajad Darabi, Shayan Fazeli, Ali Pazoki, Sriram Sankararaman, and Majid Sarrafzadeh. Contrastive mixup: Self- and semi-supervised learning for tabular domain, 2021. URL https://arxiv.org/abs/2108.12296.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE, 2013.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. 2013. doi: 10.48550/ARXIV.1302.4389. URL https://arxiv.org/abs/1302.4389.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning, 2022. URL https://arxiv.org/abs/2203.05556.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Xintian Han and Rajesh Ranganath. Core: Self-and semi-supervised tabular learning with conditional regularizations. 2021.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020. doi: 10.1109/CVPR42600.2020.00975.

Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A domain-agnostic strategy for contrastive representation learning, 2020. URL https://arxiv.org/abs/2010.08887.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Shashi Pal Singh, Ajai Kumar, Hemant Darbari, Lenali Singh, Anshika Rastogi, and Shikha Jain. Machine translation using deep learning: An overview. In *2017 international conference on computer, communications and electronics (comptelix)*, pages 162–167. IEEE, 2017.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

Appendix A. Appendix

A.1. Algorithm

In Algorithm 1 we show the pseudo code of our method.

A.2. Hyper Parameters

In Table 7 we show the values of hyper-parameters we search over. For each method we perform a grid search over relevant hyperparameter values and choose the best one according to the validation set. We use a value of $\tau_{EMA} = 0.9$ for the momentum encoder and a teacher temperature of $\tau_t = 0.04$ in Q-Match.

Parameter Name	**Search Space**	**Relevant Algorithm**
Learning rate	$[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$	All
Pre-text learning rate	$[10^{-5}, 10^{-4}, 10^{-3}]$	All
Temperature, τ	[0.04, 0.10, 0.15, 0.20, .30]	i-Mix
Corruption probability	[.3, .4, .5]	VIME, TabNet, SimCLR, SimSiam, VICReg, DINO
Student corruption	[.3, .4, .5]	Q-Match
Student temperature, τ_s	[0.05, 0.1, 0.2]	Q-Match
Queue size	$[2^9, 2^{11}]$	Q-Match

Table 7: Hyper parameter spaces for all algorithms used during training.

Algorithm 1 Pseudo Code for a Single Q-Match Training Step

```
# Create two views of the same data.
teacher_view = aug(x, teacher_corruption)
student_view = aug(x, student_corruption)

# Pass views through the model.
teacher_embed = encoder(teacher_view, ema_params)
student_embed = encoder(student_view, params)

# Normalize embeddings.
teacher_embed_norm = stop_gradient(l2_normalize(teacher_embed))
student_embed_norm = l2_normalize(student_embed)

# Compute student and teacher distributions.
teacher_logits = teacher_embed_norm @ Q.T / teacher_temperature
teacher_dist = softmax(teacher_logits)
student_logits = student_embed_norm @ Q.T / student_temperature
student_dist = softmax(student_logits)

# Compute loss.
loss = cross_entropy(teacher_dist, student_dist)
gradients = compute_gradients(loss, params)

# Update params, ema params, and queue.
params = update_params(gradients, params)
ema_params = tau * ema_params + (1 - tau) * params
Q = update_queue(Q, teacher_embed_norm)
```

A.3. Datasets and Splits

Table 8 shows the dataset splits of each experiment. For comparison with baseline methods, we attempt to stick to the splits used in the original papers. For data scaling and few-shot experiments, we create our own splits and train all methods on the same splits.

Dataset Name	**Pretext Training Set Size**	**Downstream Training Set Size**	**Downstream Test Set Size**	**Experiments**
Higgs 1%	98k	980	500k	Few Shot, Sensitivity
Higgs 5k	50k	5k	25k	Baseline
Higgs 10k	10M	10k	500k	Baseline
Higgs 100k	100k	100k	500k	Baseline
Higgs Variable Labeled	98k	{980, 9.80k, 19.6k, 98k}	500k	Data Scaling
Higgs Variable Pretext	{10k, 40k, 160k, 640k, 2.56M}	10k	500k	Data Scaling
Cover Type 1%	113400	1134	429812	Few Shot, Sensitivity
Cover Type 10%	464809	46480	116203	Baseline
Cover Type 15k	11340	11340	565892	Baseline
Adult 1%	8170	86	16281	Few Shot
MNIST 1%	57k	600	10k	Few Shot, Sensitivity
MNIST 10%	60k	10k	10k	Baseline

Table 8: Dataset splits.