Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

DeepCurl: Exploring deep curriculum learning for image classification

Author:
Bohua Peng

Supervisors:

Ben Glocker
Mobarakol Islam

Date: September 2, 2021

Contents

Abstract

Curriculum learning allows machine learning models to learn from different levels of subtasks in a meaningful order. Inspired by the fact that human learners perform better when learning from easy to difficult, researchers have mimicked this principle and designed different curriculum learning strategies. However, the definition of difficulty itself is very vague for image classification. It also remains arguable whether curriculum learning truly improves accuracy during categorization.

This project aims to answer questions including (1) what is the correlation between existing difficulty scores for image classification, (2) whether curriculum learning is useful for image classification, (3) does recently popular automatic curricula outperform handcrafted curricula.

Acknowledgements

Thank you to Dr. Ben Glocker for your guidance in this project. Thank you for providing such a interesting topic. Thank your for you supervision and advice. I learn and grow from this project.

Thank you to Dr. Wenjia Bai for allowing me to attend your weekly meetings on MICCAI2020.

Thank you to Dr. Mobarak Islam for talking to me weekly. Thank you for your technical support.

Finally, thank you to my partner, Chunyan Jia, for everything.

1 Introduction

Presenting data in a meaningful order might improve the performance of a machine learning algorithm without changing the algorithm itself. This is incredibly tempting to both academia and industry. Training models from easy data to hard ones, curriculum learning has gradually become a hotspot in recent years. However, so far, there is no accurate definition for example difficulty. Also, due to a lack of theoretical support, it remains unknown whether curriculum learning will improve performance in image classification. The objective of this project mainly consists of two parts shown in Figure 1. The first part is to investigate different difficulty scores. The second part is to test the effectiveness of different curriculum learning strategies for image classification.

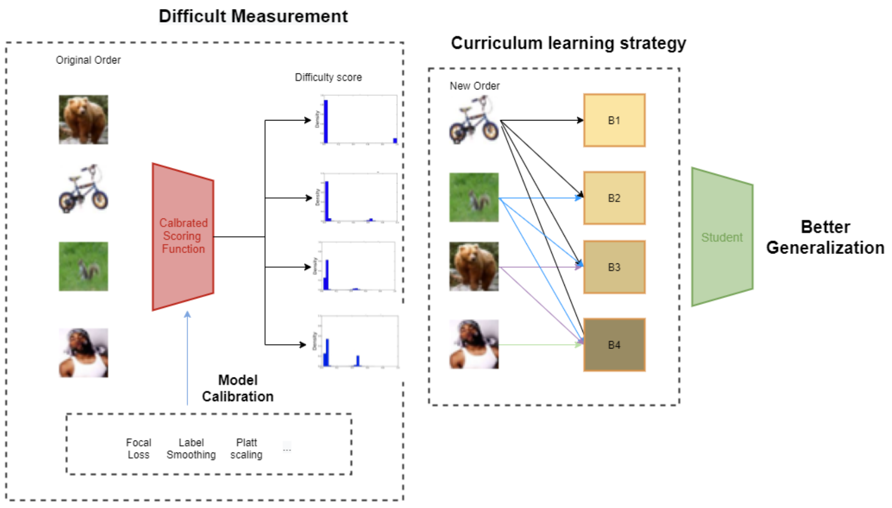


Figure 1: Curriculum learning pipeline

Example difficulty comes from cognition formed in humans' brain during human categorization. In cognitive science, human categorization has two famous models. The first one is prototype model which began with the psychological notion that humans learn rules for categories(Bruner et al., 2017). This model can quantify difficulty with the similarity between the current sample and the prototypes of candidate categories. Challenged by the work of Rosch, the first branch moves its focus from observable representations to abstract representations. In the same decade, exemplar models are proposed to create decision rules simply based on the computation with other samples. By contrast, this model measures example difficulty with the summed similarity between response caused by current stimulus and all known members of a category. Obviously, we cannot obtain human representations directly, which motivates us to look for proxies to estimate these two types of difficulties. As good function approximator, neural networks can learn high-dimensional abstract representations that resemble features used in human categorization. Therefore, the first goal of this project is a proof-of-concept demonstrating the potential of applying deep learning to difficulty measurement.

Study shows that learners will achieve better learning outcome when they learn from simple subtasks to difficult subtasks. The reason behind this has not yet been fully discovered. In recent years, deep learning methods have gained great popularity and have built strong baselines across all kinds of machine learning problems. Training deeper and more powerful models requires a large amount of data. Some of these data come from experts, but some may come from crowd annotators. The content of some data has corruption and perturbation. Some are even mislabeled. How to use a meaningful order to achieve the best learning effect has become a hot issue of widespread concern. Bengio et al. (2009) mimics the principle of human curricula and initially introduce a curriculum learning strategy that train deep learning models from easy to hard. This strategy has been widely used in image processingZhou et al. (2021), NLPKumar et al. (2019) and reinforcement learningNarvekar et al. (2020). However, Wu et al. (2020) recently challenges the effectiveness of curriculum learning strategies in image classification. Image classification is inherently noisy due to ambiguities between visually similar classes. In this context, the second goal of this project is to test whether various curriculum learning strategies are helpful for deep learning on image classification tasks. We will analyze possible problems and make corresponding improvements.

The structure of this report is as following:

1. In the second chapter, we will first introduces our benchmark CIFAR10-H. This is a dataset that reflects human perceptual uncertainty for over 10,000 images.
2. In the third chapter, we investigate different kinds of difficulty scores. We mainly focus on prediction depth and angular output margin. We measure difficulty of samples of CIFAR10-H with different metrics and evaluate their correlations.
3. In the fourth chapter, we test the usefulness of different handcrafted curricula in a considerable hyper parameter searching space. As our main attempt, we train over 900 ResNet18 models to test handcrafted curricula with different difficulty metrics. Then we finetune a SOTA classification model with CL and demonstrate the effect of CL on transfer learning. We also attempt to apply this technique to training with noisy labelled data.
4. In the fifth chapter, we test automatic curricula on a toy classification problem and extend our experience to image classification. Then we modify the SPCL framework to embed our prior difficulty into automatic curricula.

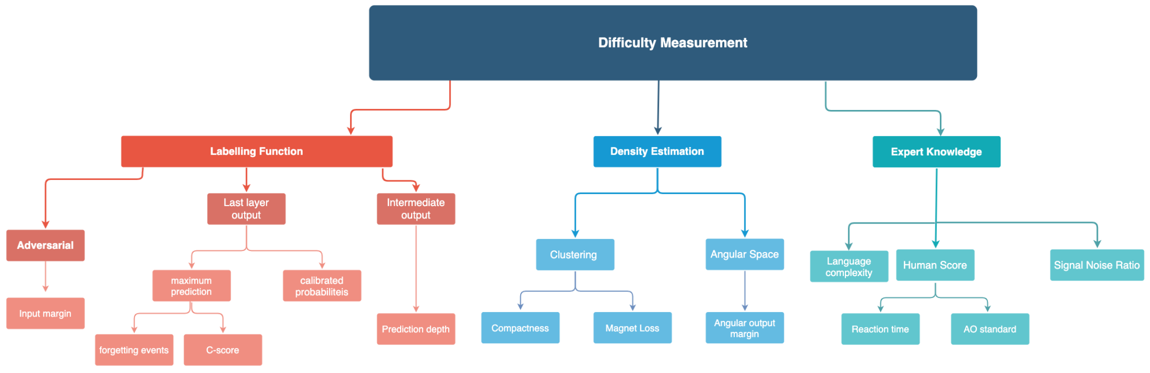


Figure 2: Categories of difficulty

2 Background

There are two principal components for a curriculum learning strategy. The first component is a difficulty scoring function that measures each sample's hardness to classification. Generally, scoring functions can be categorized into 3 classes (Figure **??**). The second component is a pacing function or a training scheduler that decides when to add in more difficult data. In this section we will first introduce different difficulty measurement. Then we will discuss handcrafted curricula and automatic curricula.

2.1 Human perceptual uncertainty

Human categorisation uncertainty is a natural difficulty score for image classification. Human uncertainty quantify the amount of hardness annotators collectively show during annotation.Although annotation can be carried out by crowdsourced annotators on the internet, it is still costly to collect enough human decisions for uncertainty quantification of a large dataset. If the dataset is too small, the variabilty per class will insufficient for accurate model evaluation.

This project will involve CIFAR-10H dataset which contains over 511,000 human predictions for CIFAR10 test set. Battleday et al. (2020) research uncertainty that people collectively show in image classification and make this dataset publicly available. These human decisions are collected from over 2570 crowdsourced annotators via Amazon Mechanical Turk. During annotation process, each participant was required to classify 200 upsampled images one at a time as quickly and accurately as they can.

2.1.1 Quality control

In this large data collecting process, there is a series of procedures for quality control. Annotators must pass the initial practicing phase and get at least 75% accuracy before entering the formal session. Label positions and presenting orders were shuffled between annotators to prevent cheating. What is more, there were attention checks where an unambiguious image was presented to each participant every 20 trials. 14 participants scored below 75% on these checks and were removed from the final analysis. With these mechanisms, Battleday et al. (2020) filter out the predictions given by these 14 participants and the number of judgments decreased from 539910 to 511,400.

2.1.2 Data Analysis

After raw filtering, we shows the distribution of the number of judgments per image in Figure 3. The range starts from 47 to 63 and the mean number is 51. 30.5% of images are annotated with 51 judgments. The difference between these numbers of judgments can be accepted. This gives more credits to human uncertainty reported by this dataset.

Then we analyse human predictions with ground truth labels. As shown in Figure

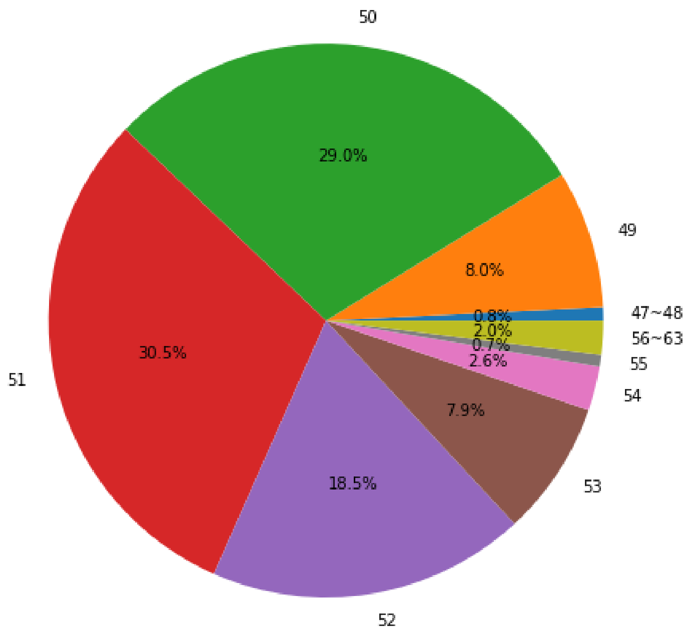
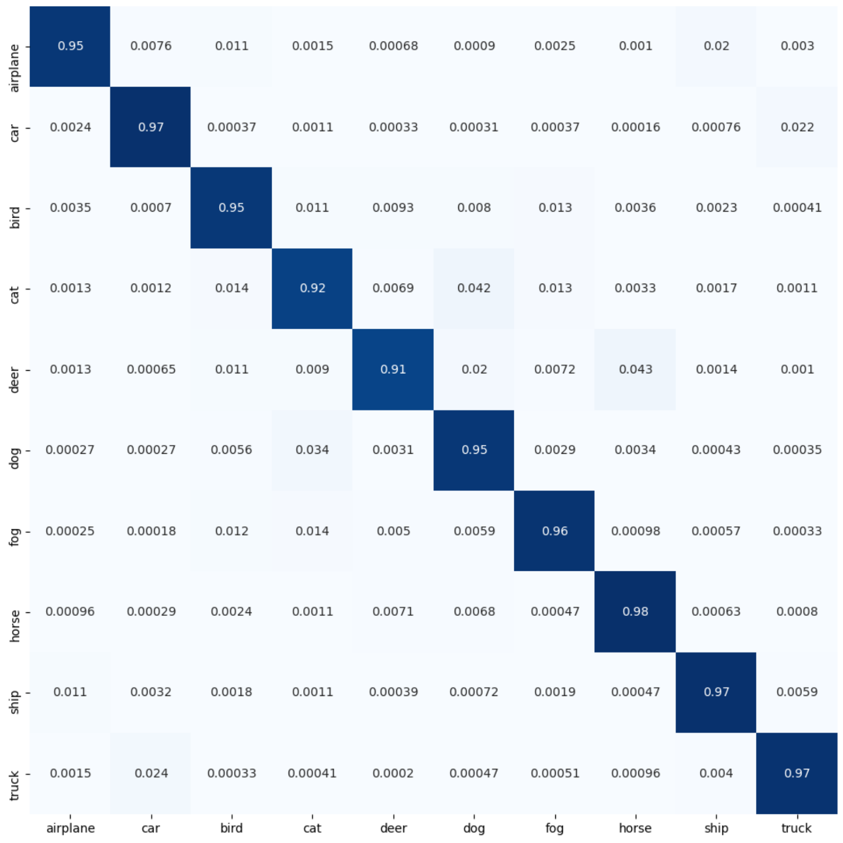


Figure 3: Distribution of number of judgments per image

4, all classes are predicted with over 0.9 precision. Human annotators are likely to misclassify samples of cat category as dog category. Human categorization also shows ambiguity between deer category and horse category. The confusion matrix shows ambiguity between classes. The matrix can be an indicator to the difficulty of each class.

In this project, we are more interested in the human uncertainty per image as a difficulty score. We show the distributions of these human scores in histograms. Each histogram is plotted with 100 bins. Therefore, we first visualize the distribution of maximum confidence of human predictions across all samples of CIFAR10-H in Figure5. Over 90% of images have consistent predictions. It shows this human guess distribution is confident about their prediction. We also compute the cross-entropy



	airplane	car	bird	cat	deer	dog	fog	horse	ship	truck
airplane	0.95	0.0076	0.011	0.0015	0.00068	0.0009	0.0025	0.001	0.02	0.003
car	0.0024	0.97	0.00037	0.0011	0.00033	0.00031	0.00037	0.00016	0.00076	0.022
bird	0.0035	0.0007	0.95	0.011	0.0093	0.008	0.013	0.0036	0.0023	0.00041
cat	0.0013	0.0012	0.014	0.92	0.0069	0.042	0.013	0.0033	0.0017	0.0011
deer	0.0013	0.00065	0.011	0.009	0.91	0.02	0.0072	0.043	0.0014	0.001
dog	0.00027	0.00027	0.0056	0.034	0.0031	0.95	0.0029	0.0034	0.00043	0.00035
fog	0.00025	0.00018	0.012	0.014	0.005	0.0059	0.96	0.00098	0.00057	0.00033
horse	0.00096	0.00029	0.0024	0.0011	0.0071	0.0068	0.00047	0.98	0.00063	0.0008
ship	0.011	0.0032	0.0018	0.0011	0.00039	0.00072	0.0019	0.00047	0.97	0.0059
truck	0.0015	0.024	0.00033	0.00041	0.0002	0.00047	0.00051	0.00096	0.004	0.97

Figure 4: Confusion matrix of human predictions

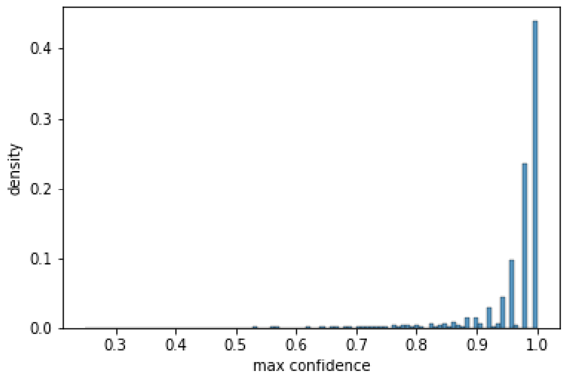


Figure 5: Distribution of maximum confidence across all samples of CIFAR10-H

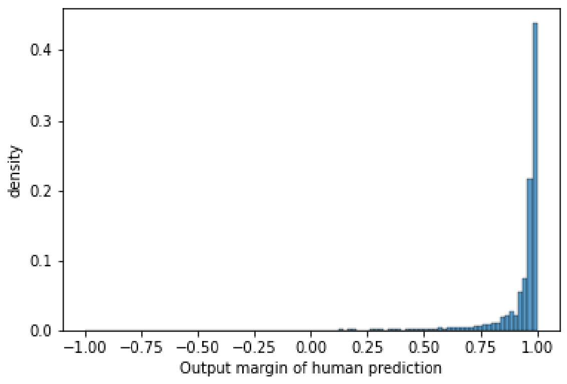


Figure 6: Distribution of output magin across all samples of CIFAR10-H

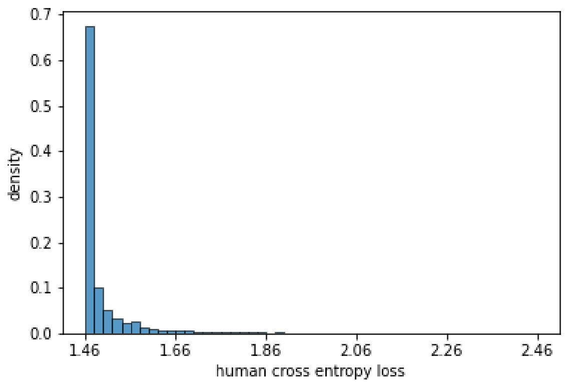


Figure 7: Distribution of cross entropy loss across all samples of CIFAR10-H

loss per image to indicate whether human predictions are right or wrong. Figure 7 shows that a small proportion of images have loss larger than 2. We compare the most confident prediction per image with the ground truth label. We find 79 images are wrongly classified by human annotators. Then we compute a quantity called output margin as the difference between the logit of the ground truth class and the second-largest logit shown in Figure 6. The advantage of output margin over loss is that there is a visually distinguishable line between correctly classified examples and wrongly classified examples. Misclassified examples have output margins lower than 0. Another benefit is that the second-largetst human prediction is taken into account.

To evaluate human prediction over possibly ambiguous examples, we randomly pick

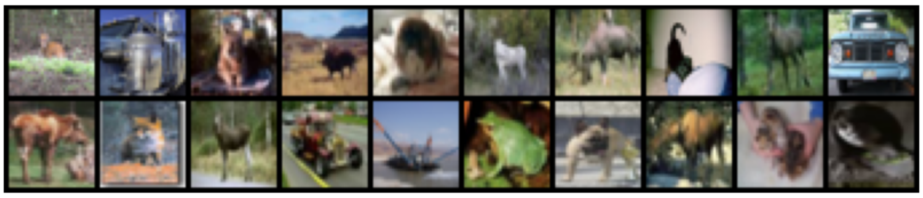


Figure 8: The first group of misclassified examples

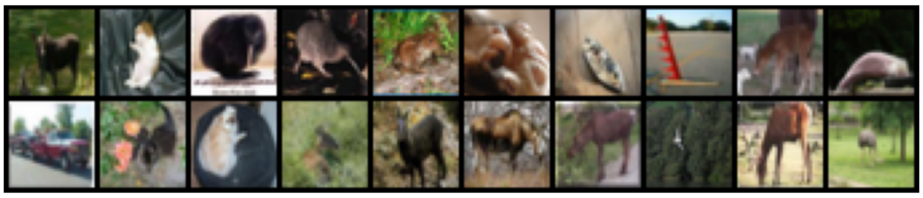


Figure 9: The second group of misclassified examples

40 misclassified examples and visualize them in Figure 8 and Figure 9. As we can see, ambiguity does exist in these images and lead to human uncertainty in image classification.

Next, we analyse the average reaction time for each image in Figure 10. For confident predictions, we also plot the average reaction time against maximum confidence in Figure 11. Generally, human annotators react fast on images with higher maximum confidence. The distribution of reaction time shows a different shape from the distribution of maximum confidence. People from different age groups can show different reaction time on the same image. Due to a lack of annotator's age information, we are uncertain if age is the key factor for the difference.

On top of these, we analyse human model calibration with human predictions. We show reliability plot in Figure 12. We find the second-largest output logits are underconfident which means output margins tend to be larger than expected. This makes human loss a more preferable choice than output margin.

To summarise, in this project, we consider images with higher cross-entropy loss per image as difficult images and smaller cross-entropy loss per image as easy ones.

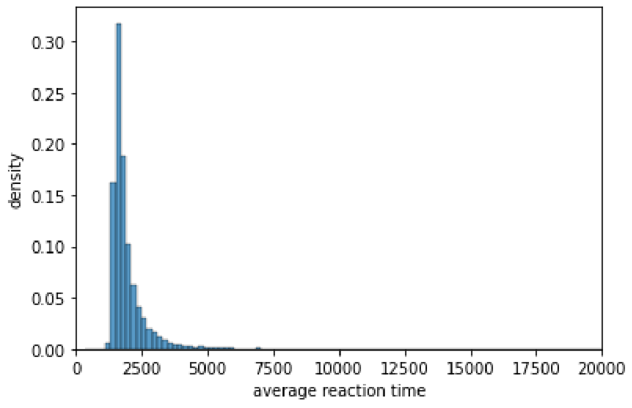


Figure 10: Distribution of average reaction time across all samples of CIFAR10-H

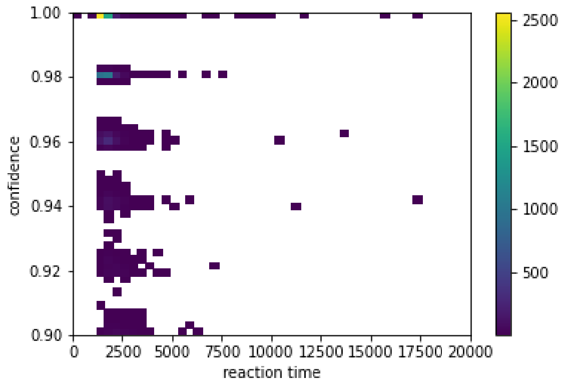


Figure 11: Average reaction time of images with over 0.9 confidence

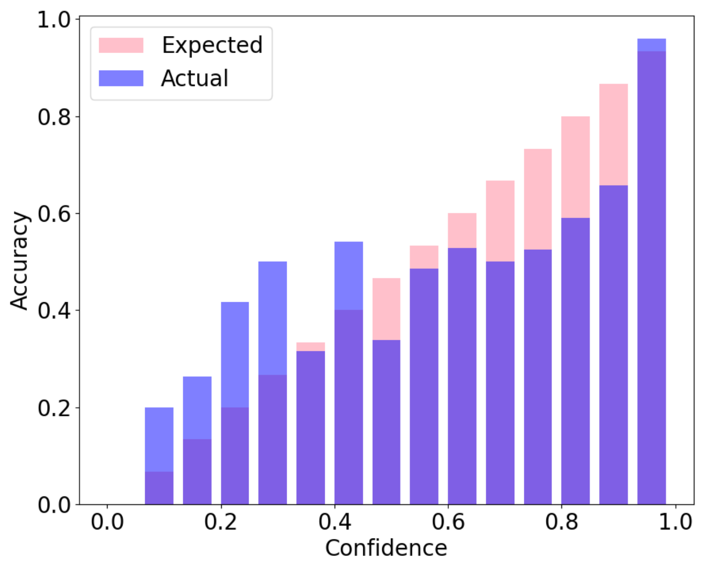


Figure 12: Reliability diagram of human predictions

2.2 Pretrained deep difficulty metrics

2.2.1 Output margin

Output margin, also known as classification margin, is defined as the difference between the logit of the ground truth class and the second-largest logit. Generally, the deep learning model exhibits larger output margins for easy samples. Output margin is widely used as baselines in difficulty measurement literature (Toneva et al., 2019; Baldock et al., 2021). Soudry et al. (2018) demonstrate that overtraining with negative loglikelihood give rise to large output margins. (Baldock et al., 2021) refer output margin as "local simplicity". They intervene the output margin of a network with Hinge Loss to control difficulty score they propose. An advantage of output margin is its flexibility. This quantity can be easily modified to work with other mechanism. For example, Toneva et al. (2019) creates the notion of misclassification margin by dividing the output margin with forgetting events.

Although output margin is easy to compute, this metric is sensitive to model calibration and architectures. To make this metrics more stable, it is a common practice to train an ensemble and use the average output margin.

2.2.2 Forgetting events

Toneva et al. (2019) defines a forgetting event as an example's accuracy decreases between two consecutive updates. Put it simply, "forgetting" signifies that an instance is correctly categorized at step t but wrongly categorized at step $t+1$.

From the definition, it is too costly to monitor forgetting events as it requires computing the predictions for all data points of the current training set. To practically compute example forgetting events, they lower bound true example forgetting events in a mini-batch gradient descent fashion. Specifically, general example forgetting events can be computed in two main steps as follows. One first trains a single classifier on a given dataset and record the statistics (loss, accuracy, misclassification margin) for each example when they present in the current mini-batch. In the second step, one computes number of forgetting events per example and sort examples by forgetting counts.

Samples that are never learnt are given the maximum number of forgetting events of the current dataset. They analyse the metric on its stability across different random seeds, numbers of training epochs and different architectures. Using Spearman rank correlation, they empirically proves the metric's consistency across five random seeds, 75/200 training epochs and various CNN models.

As for the weakness, considering that accurate classification could be produced by chance mostly in the case of a small number of classes, "forgetting" may be problematic. In this regard, it is hard to determine whether a "forgotten" example was learned initially. To be more specific, Toneva et al. (2019) quantify "chance" forget-

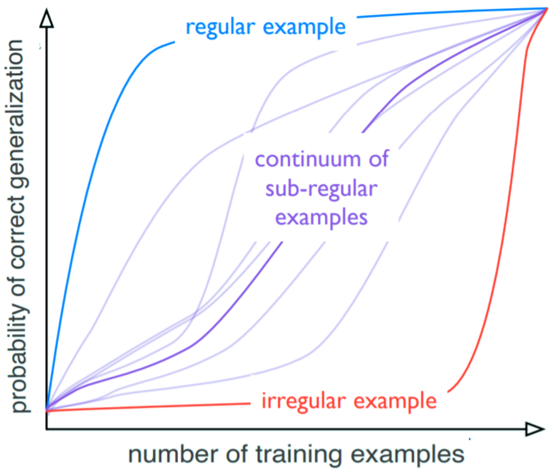


Figure 13: Measuring structural regularities of training samples with C-score

ting by analysing the forgetting rates with random gradient updates as follows. The "base" classifier is first cloned into a new "clone" classifier with the same random weights at the beginning of training. Then, the shuffled backward gradients of the base classifier are copied to the clone one. Finally, the forgetting events reported by the clone classifier are computed in the exact two main steps as the base classifier. With a forgetting events histogram, they empirically show CIFAR10 training examples can be randomly forgotten at most twice. This means the unforgettable (easy) samples are plausible and difficult ones are likely to be not.

2.2.3 C-score

C-score (Jiang et al., 2020) model structural regularities of the training data by estimating the probability of correct generalization for a specific validation set. According to their observation, deep learning models learns simple functions from regular samples and memorize complex functions from irregular samples. As a result, simple functions generalize well and achieve high accuracy while complex functions generalize poorly and have lower accuracy. As illustrated in Figure 13, it takes less regular samples to get a good generalization but it takes a lot more for irregular ones.

Since samples of different regularities are mingled, the only way to compute this quantity is through Monte Carlo sampling. As shown in Figure 14, one randomly samples a subset of training set and compute the validation accuracy with the remaining data. Then this score is given equally to the train split as regularities of these samples. In their paper, they train 2000 models to perform Monte Carlo estimation. Their work is the first of its kind and can be theoretically useful for curriculum learning, active data selection and outlier detection.

However, there exist several problems for this novel difficulty metric. First, it is very difficult to find the optimal ratio between train split and test split. With a larger test split, the evaluation becomes more precise, but it also means more runs to evaluate all data. With less training data, model also suffer higher risk of overfitting and pre-

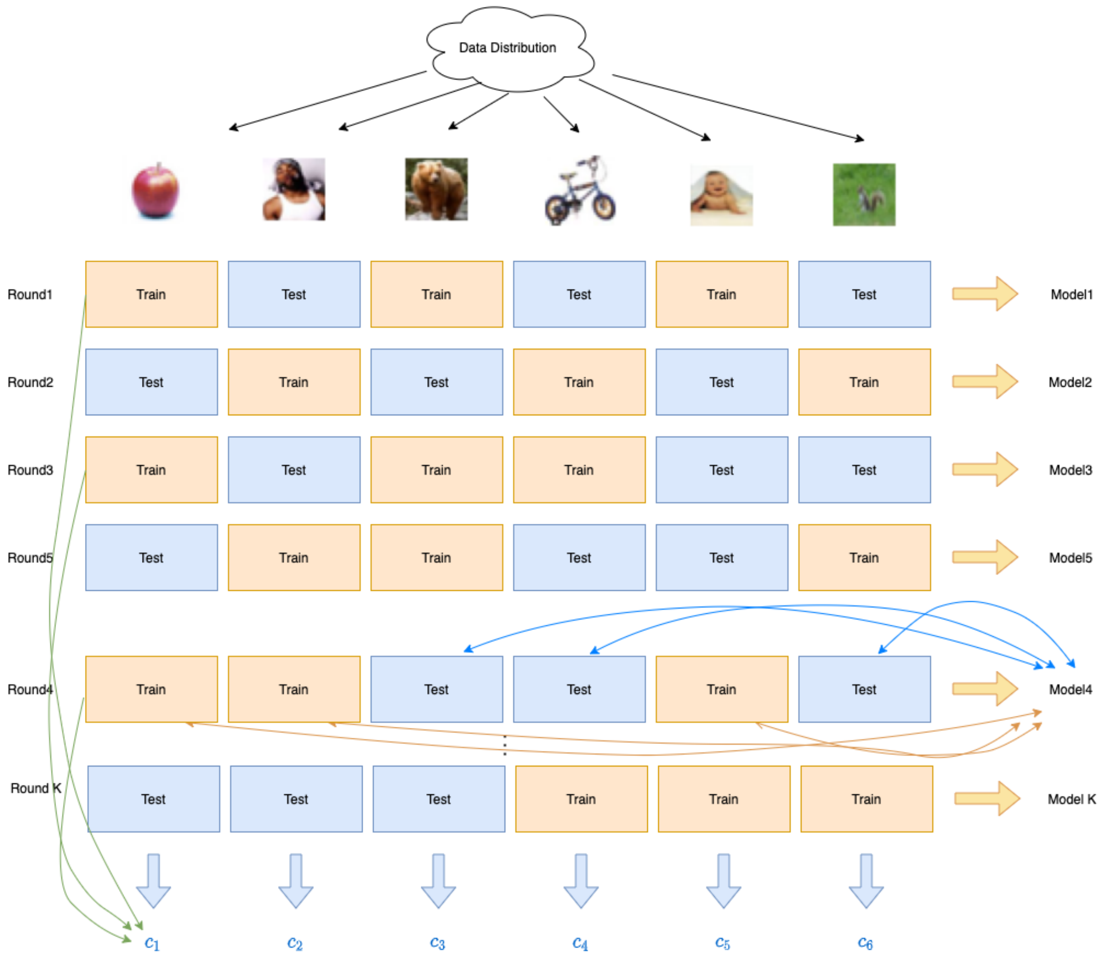


Figure 14: Measuring structural regularities of training samples with C-score. To implement this, one runs multiple runs and splits the training and validation split randomly in each run. This ensures both splits come from the same distribution. Then, one trains a model on the training split and test it on the validation split. Next, one computes the average accuracy and assign this regularity score equally to examples in the training split. After hundreds of runs, one averages across all runs and gets the regularity scores for all samples.

dict meaningless results. Currently, the optimal ratio is found by grid search which is infeasible for most people. Second, Monte Carlo sampling requires so much computational resource that makes computing C-score infeasible for most people. Third, there is a shortage of theoretical support justifying C-score relates to regularities. One way to test C-score is through visualizing examples along with their C-scores. In this project, we opt to test C-score with curriculum learning.

2.2.4 Prediction depth

So far, we have discussed difficulty scores defined by the labelling functions of neural networks. These metrics mainly focus on the output logits of the final fully connected layer and fail to fully utilize the hidden representations from the intermediate layers. Recent findings have shown deep models learn easy data and simple functions first Stephenson et al. (2021). And deep neural network layers converge from the input layer towards the output layer Raghu et al. (2017). Therefore, one can measure example difficulty intermediate convergence with k-NN classifiers plugged in the intermediate layers. Baldock et al. (2021) introduce a difficulty score called prediction depth that utilize the natural clustering effects of deep learning models.

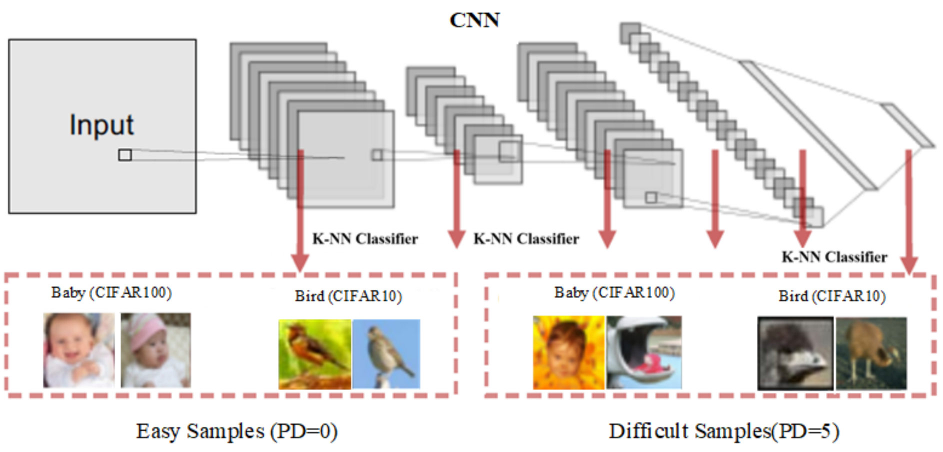
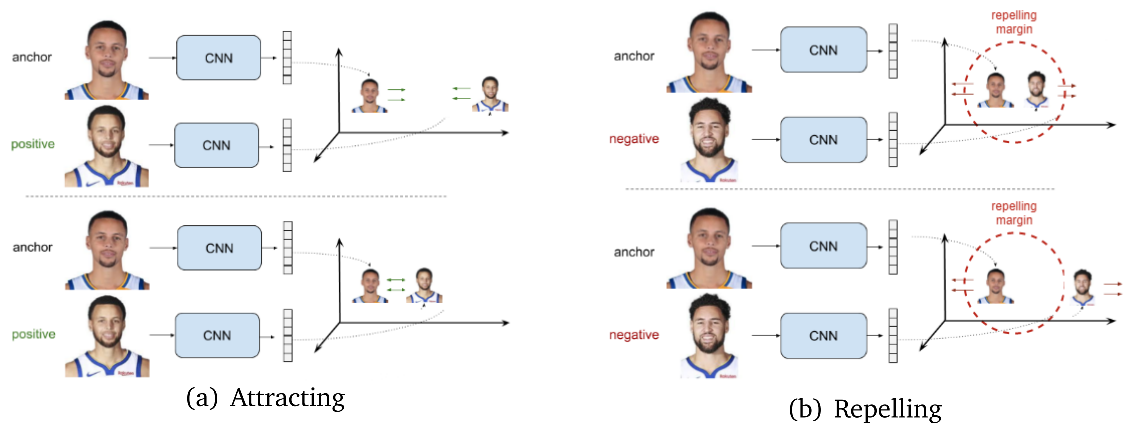


Figure 15: the mechanism of prediction depth

DDBased on discriminative hidden representations, prediction depth defines example difficulty as the earliest layer where all the subsequent intermediate predictions converge to a fixed label Baldock et al. (2021). As shown in Figure, one can place multiple k-NN probes after certain layers in a neural network. With the labels of a support set, each k-NN classifier outputs a prediction determined by the distance between features. If the predictions from a layer and its successors converge to the same label, one can record the number of the first layer where convergence begins. Based on this score, input data are stratified into several difficulty levels. With a single model, an example's prediction depth can only be an integer. One can split the input data randomly and train an ensemble with the same architecture in a supervised learning fashion, and then take the average prediction depth across models. This extends prediction depth to continuous and facilitates more precise data scheduling.



(a) Attracting (b) Repelling

Figure 16: Contrastive loss pulls similar faces together and push dissimilar faces away

2.2.5 Deep metric

As a metric, difficulty score can be directly measured on abstract representations learnt by deep neural network. This idea connects difficulty measurement to deep metric learning.

Generally, deep metric learning (DML) methods fall into two major categories depending on the type of supervision available. Weakly supervised DML is designed for data that are in pairs, triplets or n-pairs. With a form of contrastive loss, these weakly supervised machine learning methods attract similar samples and repel dissimilar samples. As shown in Figure 16, traditional contrastive encourage pulling positive samples together while pushing negative samples out of a margin. Recently, infoNCE loss is introduced to modern contrastive learning, and it allows efficiently attracting and repelling n-pairs. Interestingly, supervision comes in the form of data augmentation and these methods are fancily named self-supervised learning.

However, the contrastive loss is unsuitable for supervised learning. This is mainly because the methods mentioned above need to intensively sample from classes to generate pairs in the data pre-processing stage. Another problem is the attracting and repelling are at the instance level, so convergence is often slow due to a lack of global structural information. By contrast, supervised deep metric learning methods efficiently find a target space by pulling samples of the same class together while pushing samples of different classes away. One can use different forms of distance to describe similarity in the latent space. If one believes the latent space is a Euclidean space, he can take the average of representations of the same class/cluster as class/cluster centroid. To maximize discrimination between classes, Center loss Wen et al. (2016) adds intra-class variances to cross-entropy loss. This loss function encourages the representations of the same class to be closer.

Some researchers believe that category labels are not accurate enough to describe similarities. So they introduce clustering and use the pseudo-labels generated by clustering for supervised learning. Instead of retrieving a class of samples, Magnet loss (Rippel et al., 2016) applies a clustering algorithm on representations and assign samples with fine-grained cluster indices. Then the algorithm pulls the samples of the same cluster together while pushing samples of different classes apart. As a result, the intra-class variance can be controlled in a proper range and the diversity is maintained. SPL-ADVisE uses the Magnet loss as a difficulty score and applies self-paced learning on image classification.

SpCL learns a joint representation space for domain adaptation with contrastive loss and curriculum learning strategy. They have labelled data on the source-domain and the representations are learnt in a supervised learning fashion. As for target domain, there is no label and training is done with self-supervised learning. If one only apply contrastive learning at instance-level, he will not get a high accuracy due to a lack of global structural constraints. Their idea is to perform DBSCAN on the latent space twice with a changing hyperparameter. This will give each cluster an IoU.They define and control the compactness of each cluster with this IoU. Samples belongs to a compact cluster are considered as easy samples and samples belongs to a loose cluster are considered as difficult samples. They train easy samples with cluster pseudo-labels and difficult samples with contrastive loss. This allows them to achieve SOTA performance in person re-identification task.
Alsharid et al. (2020) apply curriculum learning to multi-modality by measuring the similarity between the embeddings of video frames. The difficulty can be measured by the Euclidean distance between the current frame and all others in the dataset. When the distance is small, the new frame is considered as an easy sample and added into training.

2.2.6 Other deep learning scores

Jiang et al. (2019) introduces the adversarial input margin, and linearly approximate this quantity with a ratio between output margin and adversarial gradients. This denotes the smallest norm for an adversarial perturbation to change the model's class prediction.

Based on the background study, we can summarize two hypothesis for difficulty measurement. First, an ideal difficulty scoring function needs to accurately reflect true example difficulty with regard to downstream tasks. The latter property implies the hardness of defining difficulty due to its multiple definition even in the same downstream task. For example, in the well-known linear regression with clean data, if we have an optimal linear regressor with parameters w^* (least square solution), the example difficulty can be intuitively defined as distance between prediction $\hat{y}$ and y. Nevertheless, example difficulty can be defined from optimization perspective when the problem size is large.

Second, a good difficulty scoring function should have certain mechanisms to enforce smoothness. This property will facilitate training scheduler design. This hypothesis can be easily explained with an extreme case. Let us consider measuring example difficulty with static binary labels. Here, “static” means example difficulty can only be evaluated once as either 0 (easy) or 1 (hard) and cannot be changed afterwards. In this situation, the choice of scheduler is limited to a step function that decides when to add all difficult samples in. This strategy is paradoxical as samples should not be equally important in curriculum learning. If one insists on using binary /one-hot predictions to measure difficulty, then the difficulty measurer must include mechanism to smooth the binary / one-hot outputs.

2.3 Curriculum learning methods

2.3.1 Predefined curricula

Paced learning (PL) manually designs a curriculum with precomputed difficulty scores estimated by difficulty measurers. Specifically, PL first sorts training data from easy to hard with difficulty scores. Then the method uses a scheduler, also known as pacing function, to decide when to add in more difficult data. With a learning rate annealing strategy, the method dynamically scales down the updates from difficult samples.

Generally, scheduling strategies fall into two forms, discrete scheduling and continuous scheduling. Discrete schedulers split data into a predefined number of buckets and add in more difficult data when it detects performance stagnation.

Algorithm 1 Discrete training scheduler

```
Input: S:training dataset; M: difficulty measurer;
Output: $W^*$: optimal model parameters
  $S, order \leftarrow sort(S, M)$;                        ▷ sort S from easy to hard
  $[S^1, S^2, ...S^T] = S$
  $S^{train} = \phi$
  for t = 1,2, ... T do
      $S^{train} = S^{train} \cup S^t$                     ▷ add in more difficult samples
      while not stagnate train for k epochs do
          $train(S^{train}, M)$                            ▷ apply normal mini-batch training
      end while
  end for
  return $W^*$
```

Baby step schedulers are the most widely adopted discrete scheduler (Bengio et al.,

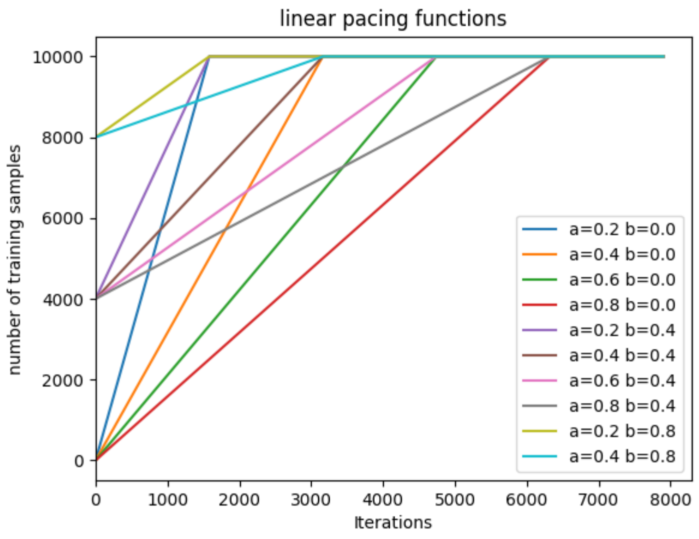


Figure 17: Linear pacing functions

2009; Wang et al., 2021). As shown in Algorithm 1, this scheduler first puts sorted training data S into buckets S^t from easy to hard. Then training starts from the easiest bucket S^1. Next, the algorithm performs standard mini-batch training for k epochs until the model converges on the current subset. Then the algorithm adds in next difficulty level bucket S^t into the training subset. The algorithm repeats these two steps until all buckets are added and exploited. Finally, training either stops or continues with all data for several epochs, and outputs the model parameters. The baby step scheduler usually shuffles data within each bucket and then sample mini-batches for training instead of using all data. Baby step schedulers are simple and build a strong curriculum learning baseline(Spitkovsky et al., 2010).

One-pass is another discrete scheduler that is less used than the Baby step(Wang et al., 2021). The difference between the two is in updating. One-pass scheduler switches to the next difficulty level bucket while discarding data learnt before. This leads to catastrophic forgetting which is often reported in continuous learning literature. The limitation of a discrete scheduler is that the performance is sensitive to the number of buckets and designers have to tune this hyperparameter manually.

As an alternative, continuous schedulers can add in hard data at every iteration or epoch with pacing functions. Pacing functions are a group of monotonic non-decreasing functions. As shown in Figure 17, b denotes the initial percentage of data when the training starts. aT denotes the iteration where all data are added.

There exists a variety of pacing functions in CL literature. However, we focus on three types of pacing functions in this project. Shown in Figure 17, linear function are the most straightforward pacing function that adds data into training subsets from easy to hard linearly. Root function, shown in Figure 19 is concave and spends more time on difficult samples than a linear function. By contrast, the quadratic function, shown in 18 spends more time on easy samples.

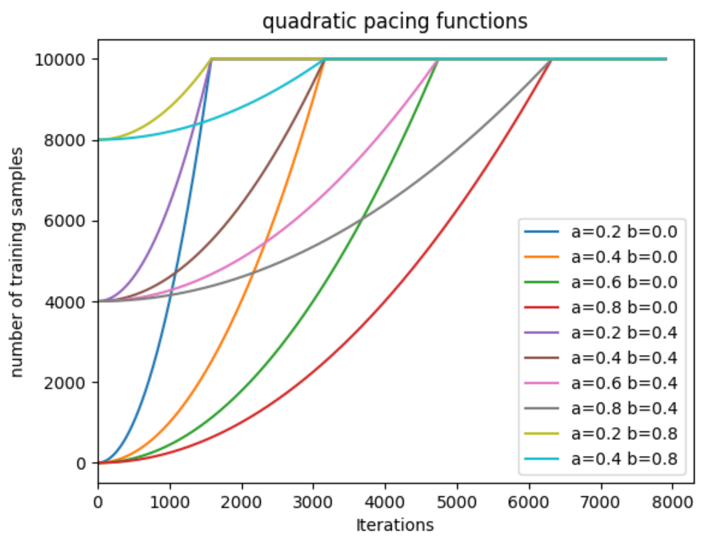


Figure 18: Quadratic pacing functions

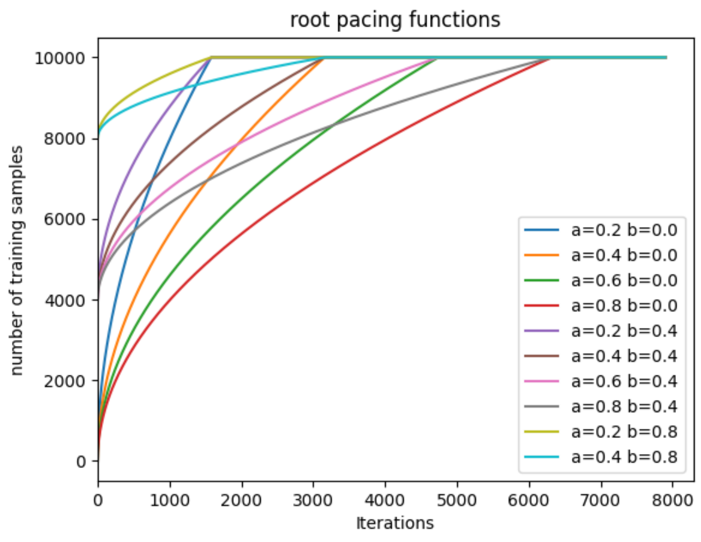


Figure 19: Root pacing functions

Although the data adding process has become easy in continuous schedulers, the relative improvement to other schedulers is not drastic. Wu et al. (2020) adopts a variant of the continuous scheduler in their work and shows curriculum learning improves model convergence and the performance of training with noisy labelled data. However, their conclusion that curriculum learning does not help standard training remains arguable because their data loaders shuffle samples across data buckets which severely damages the global difficulty structure[1]. We believe ties are broken when shuffling the original dataset.

2.3.2 Automatic curricula

Kumar et al. (2010) initially designs a self-paced learning (SPL) strategy inspired by the learning process of humans that gradually incorporates easy to complex data into training. This learning strategy considers an automatic curriculum as a weight vector and update this weight vector with the current model's feedback. Unliked fixed handcrafted curricula, the adaptive design makes the curricula learnt by SPL evolve with the current model. The loss functions of SPL generally follow the paradigm proposed by Kumar et al. (2010) which consists of a weighted loss term and a regularization term for the weight vector. SPL alternatively updates model parameters and curriculum weights with batch gradient descent (Jiang et al., 2018). The regularisation terms are often convex functions that allow the weight vector to have closed-form solutioin. Since curriculum get updated independently, it works as an off-the-shelf tool to other machine learning algorithms Ge et al. (2020).

SPL has achieved good performance on computer vision and NLP tasks (Cirik et al., 2016); (Zhang et al., 2018)).Pentina et al. (2014) apply self-paced learning strategies to multi-task learning where SVM learns to perform a sequence of tasks from easy to hard. SPL has also becomes a hot spot in Reinforcement learning (Klink et al., 2020); (Kumar et al., 2019); (Narvekar et al., 2020)).
For image classification, SPLD (Jiang et al., 2014) combines the SPL algorithm and clustering to consider the diversity of samples during image classification. MentorNet (Jiang et al., 2018) estimates curriculum weights with RNN models use mini-batch SGD for better convergence.

Self-paced curriculum learning (SPCL) develops a paradigm where teacher and student can learn collaboratively (Jiang et al., 2015). It bridges the gap between handcrafted curricula and automatic curricula. In SPCL, the teacher can propose an outline for the curriculum that offers key guidance in the initial stage. According to the teacher's advice, students can learn specific knowledge at their own pace. To achieve this, precomputed difficulty scores are embedded into SPL as constraints. When the constrained optimization problems are convex, one can still compute the weight vector analytically. By contrast, if non-convexity exists, one must find the solution

[1]Readers may check their implementations (Wu et al., 2020)

for curriculum parameters numerically. ulty scoring function should be accurateenough to reflect example regularity or ambiguity. By "accurate enough", wemean the scores are distinguishable for sorting. Therefore, it is crucial to In this project, we opt to test the performance of SPL on image classification and the advantage of using mini-batch SGD over batch GD.

2.4 Confidence and model calibration

Confidence is the probability predicted by the classifier of a specific class. Overconfidence and underconfidence are measured by model calibration. A highly accurate model is overconfident for a class if the class predictive probability is higher than the accuracy of that class. Model calibration plays two roles in this project. First, model calibration can be used to rescale neural network's outputs for difficulty measurement. An efficient difficulty measurer prefers output probabilities over maximum predictions. To make output probabilities plausible for difficulty measurement, the model should be calibrated so that the probability corresponds to the proportion of samples being classified correctly. Second, when the dataset has noisy labels, an intuitive solution is to set a threshold on example loss and filter out noisy samples. This requires precise output probabilities.

One way to measure the degree of calibration is to compute Expected Calibration Error (ECE). For binary classification, the method separates data into different bins according to their probabilities of positive class and calculate the accuracy of samples in those bins. Then the method computes the absolute error between the average confidence and the average accuracy for each bin. Finally, the method does a weighted average of these absolute errors across all bins. Reliability diagram**?** provides a popular visualization with equally-spaced bins and plots the confidence against accuracy. The plot of a well-calibrated model should show a diagonal pattern. For multi-class classification, ECE takes the maximum confidence of all probabilities. Classwise ECE replicates the previous binning strategy for each class independently and takes an even averaging across different classes at the end. Recent calibration metrics put more focus on the most confident predictions or predictions under thresholding because they are used more often. Specifically, Nixon et al. (2019) loops through different classes and first sorts the predictions based on confidence. Then the method thresholds small confidence predictions out. Then the method finds adaptive bin intervals so that each bin contains a similar number of predictions. Finally, the method takes a weighted average across all bins to get TACE.

Recalibration can be easily applied with post-processing. Platt scaling is a test-by-time parametric method. To apply Platt scaling, one simply fixes the weights of a model, takes the output logits, and trains an extra linear layer with a hold-out validation set. This is called the matrix scaling version of Platt scaling. Guo simplifies this matrix with a single scalar T and their method is known as temperature scaling. They also show that neural networks can achieve higher accuracy in the sacrifice of worse calibration. TACE (Nixon et al., 2019) and (Guo et al., 2017) evaluate several calibration methods including variants of Platting scaling, variants of histogram binning, isotonic regression and Maximum Mean Calibration Error. It has been shown that variants of Platt scaling especially temperature scaling achieve a higher ECE.

Implicit model calibration methods have recently become popular in image classification(Müller et al., 2019) and image segmentation(Islam and Glocker, 2021). Despite a lack of theoretical support, these studies show strong evidence that label smoothing implicitly improve generalization and calibration. A plausible explanation is that training with soft labels enforces data to lie in tight equally separated clusters in the latent spaceMüller et al. (2019).

3 Difficulty Metrics

3.1 Model calibration

As discussed in section 2, deep learning models need to be calibrated before difficulty measurement. In this subsection, we evaluate different types of model calibration techniques to understand how to wisely choose them for different metrics.

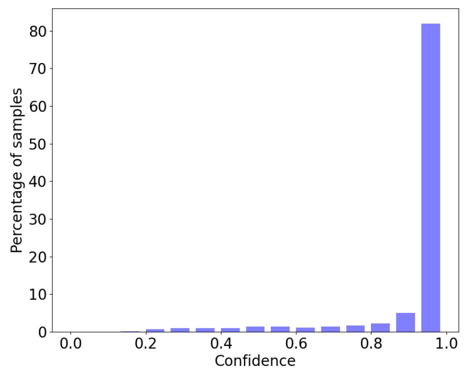
We first randomly split the CIFAR10 with 90 percentage of data for training and 10 percentage of data as hold-out validation set. We train a ResNet18 model on the train split for 350 epochs to make it overconfident over its accurate predictions. The weight decay parameter equals $5e-4$. We use a cosine annealing learning rate scheduler that starts from 0.1 and ends with 0. We use the standard data augmentation technique for CIFAR datasets that includes random crop with padding, random horizontal flip and normalization. Then we use the validation set to implement model calibration. We apply Temperature scaling by dividing the output logits of the neural network with a temperature parameter. Next, we freeze the model parameter and train this single temperature parameter on the validation set for 10 epochs with an Adam optimizer whose learning rate starts from 0.001. After it converges to 2.50, we record ECE and ATCE with binning. We have done a grid search over this temperature parameter and get similar results. We plot the reliability diagram along with bin strength in Figure20 For Brier Score we keep the same experimental setup except changing our loss function to an L2 loss. This gives us a temperature of 1.04.

However, for label smoothing, we retrain a ResNet18 model from scratch with soft labels and the same configuration. For soft labels, we decrease the confidence of the ground truth class from 1 to 0.95 and split 0.05 to equally to other classes, and then we minimize the negative loglikelihood with soft labels. We also switch the loss function as Focal loss and set γ as 2 to prevent overconfident predictions. The reliability diagrams of Label smoothing and Focal loss are shown in Figure21 and Figure22.

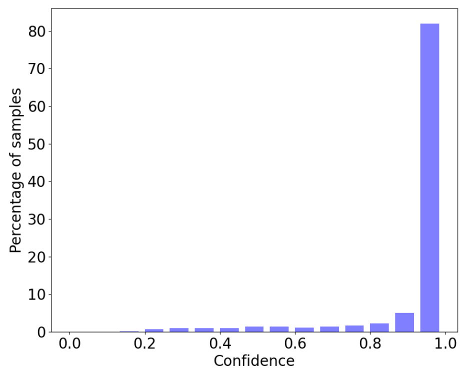
We compare the results of different model calibration techniques in Table 1

Table 1: Calibration methods' performance on noisy labelled data detection

	Post	ECE	ATCE	AUC($\epsilon = 0.2$)	AUC($\epsilon = 0.4$)
Human	No	0.0375	0.0067		
Platt Scaling	Yes (2.50)	0.0275	0.0049	0.979	0.979
Brier Score	Yes (1.04)	0.0252	0.0044	0.984	0.986
Focal Loss (γ=2)	No	0.0338	0.0046	0.984	0.984
LS (γ=0.05)	No	0.0306	0.0062	0.021	0.947

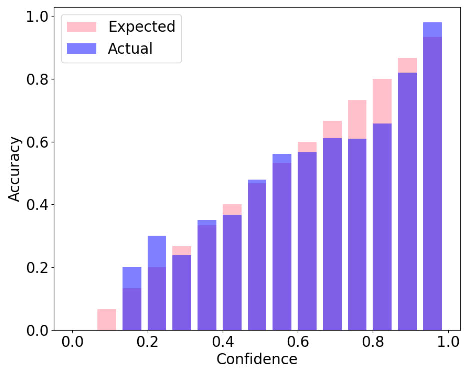


(a) Reliability diagram

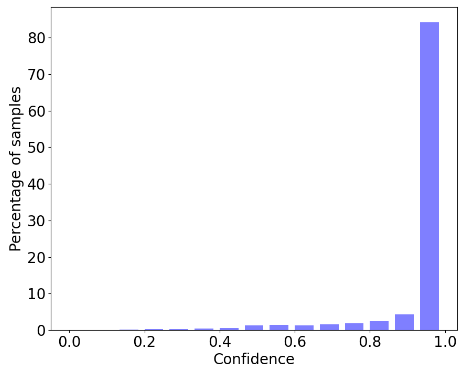


(b) Bin strength

Figure 20: Reliability diagram and bin strength of Temperature scaling

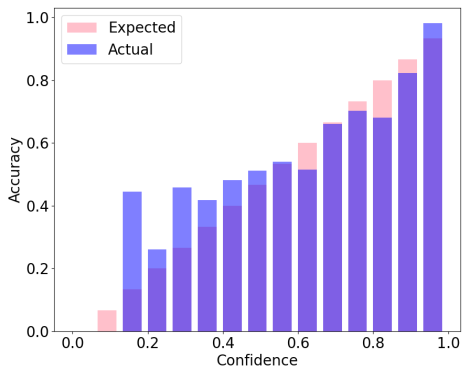


(a) Reliability diagram

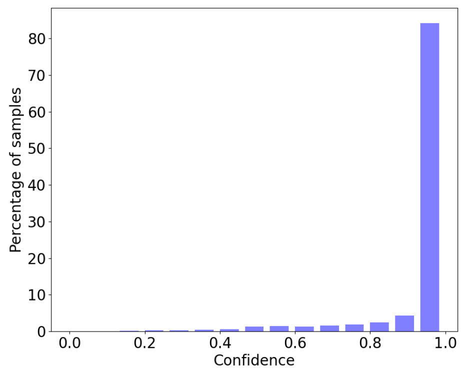


(b) Bin strength

Figure 21: Reliability diagram and bin strength of Label smoothing



(a) Reliability diagram



(b) Bin strength

Figure 22: Reliability diagram and bin strength of Focal loss

3.2 Prediction Depth

To distinguish different forms of difficulty, we calculate two prediction depth values for each data point. The first group of PD evaluate the specific training split used to train the model, and we define them as train split PD. These quantities are likely to be biased estimation of true difficulty as the support set are used for training. The other group of prediction depth values use the validation split as their support set. The calculation is similar with cross validation which means they are unbiased. We run multiple runs. Each run gives us train splits evaluation and test split evaluation. Averaging across these runs leads to two group of more stable prediction depth values.

3.2.1 On two dimensional difficulty

Baldock et al. (2021) did not publish their code, and we write down our implementation are as follows. The algorithm is as follows.

1. To begin with, we place k-NN probes in DNNs to get intermediate predictions. In MLP, probes are constructed after dense operations and the softmax. In CNN, probes are placed after convolution and before the pre-activation.
2. In each run, we randomly split the dev set into the training split and validation split and hold-out with 8:1:1. We train a ResNet18 model on the train split with standard i.i.d. training. We freeze the model parameter and use it as a feature extractor for the following steps.
3. To evaluate each point in the validation split with train time PD metric, we extract and normalize the intermediate features of data in the train split. We calibrate k-NN probes with Temperature scaling before classification. We create a search grid for k-NN temperature parameters and compute ECE on the hold-out set. Then we apply k-NN classification and recorder intermediate labels. Whenever we classify a data point with k-NN, we use the specific training split used to train that model as the k-NN support set. In this way, we do not need an extra support set. We compute PD by reversing the list of k-NN predictions and record the depth when misalignment occurs.
4. We compute PD for each sample in the train split following the similar way as in the third and fourth steps. The only difference is when calculating the PD for a point in the training split, we leave this point out of the k-NN support set. Specifically, we find the k+1 nearest neighbours from the training split and then delete the closest.
5. Finally, to get stable example difficulty, we repeat 2-5 for multiple runs with different random seeds. We take average over these runs for train split PD and test split PD separately.

After training 64 ResNet18 models on CIFAR100 , we plot the train split prediction depth against the validation split prediction depth in a two dimensional histogram as Figure. We visualize the samples in the 4 corners of this 2D space. Easy data are with low PD_{train} and low $PD_{validation}$. They are typical examples of their classes. These samples can be easily classified, and their predictions match ground truth labels all the time. Ambiguous without labels data (Low PD_{train}, Low PD_{train} often include visual contents that both look like their ground truth classes and another class. These features can be informational and reside on the decision boundary of classifiers. Ambiguous samples (High PD_{train}, High $PD_{validation}$) suffer from low resolution, low contrast or object occluded. The randomness of training can often affect their prediction. Data look like a different class (High PD_{train}, Low $PD_{validation}$) include confusing features of another class. These samples are disconnected from other samples of the same class. They can be memorized during training but are always misclassified during validation.

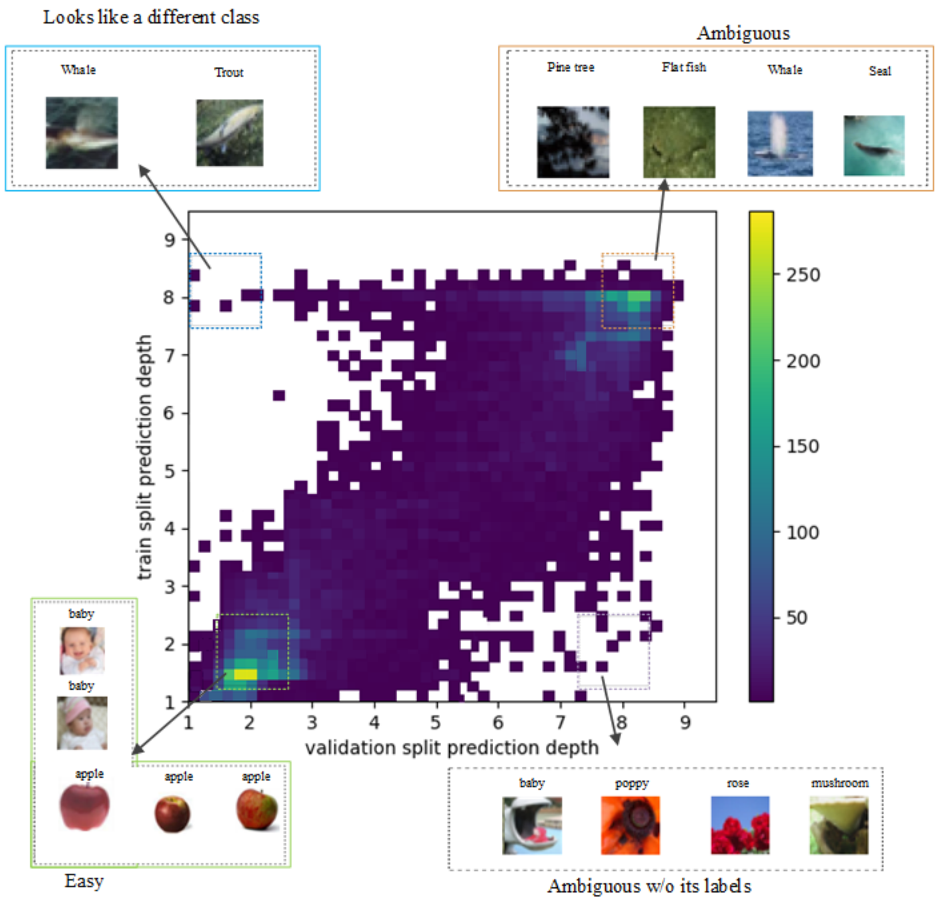


Figure 23: 2D histogram of train split prediction depth and test split prediction depth CIFAR100 images

Additionally, we choose an equal number of samples of these four difficulty forms for further investigation. For each layer, we plot the mean k-NN probe confidence of the ground truth class against accuracy. This is another visualization of how an ensem-

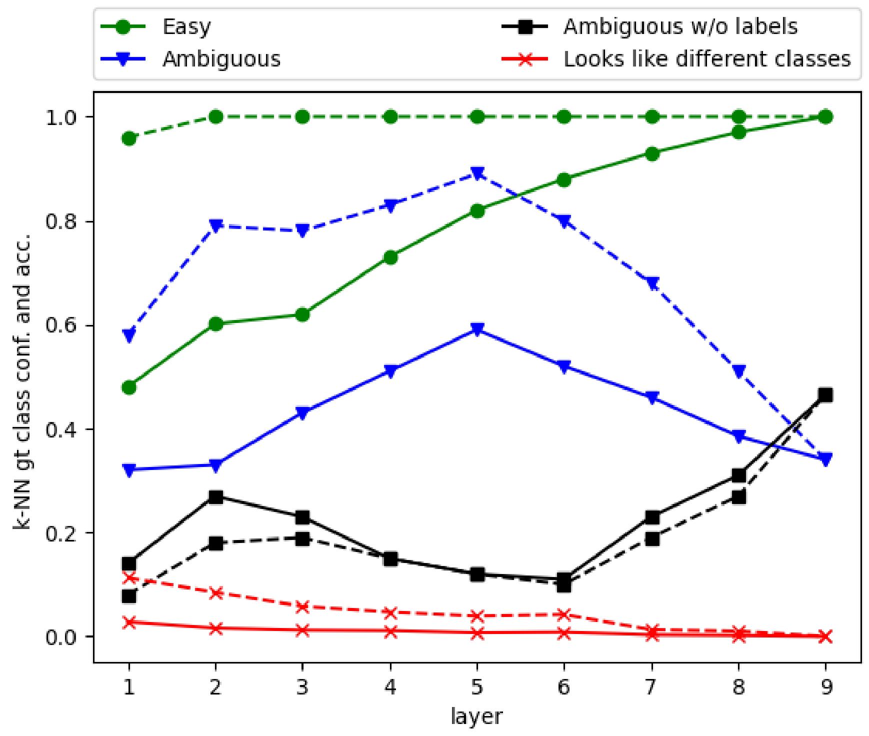
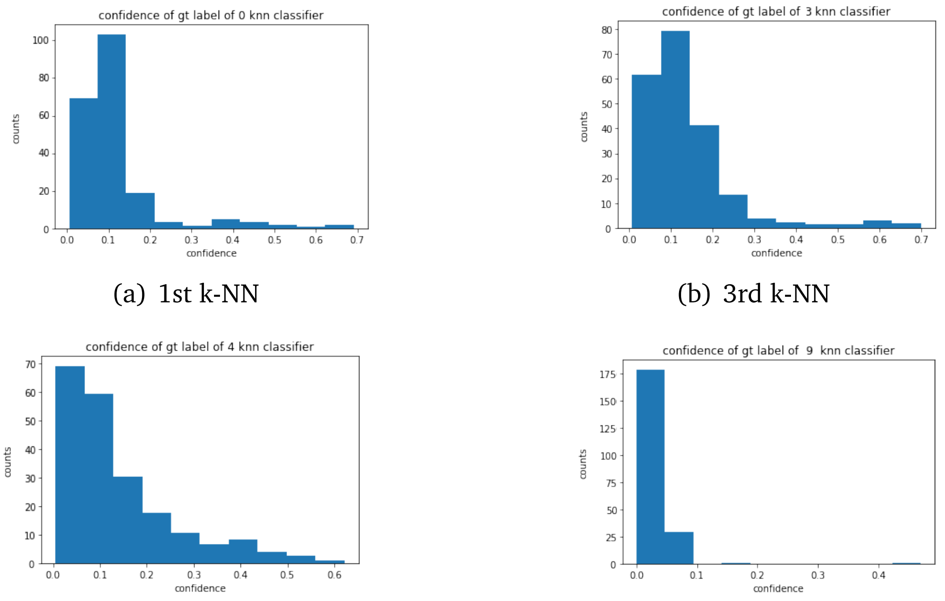


Figure 24: k-NN probe's confidence of the ground truth class against accuracy. For samples that are ambiguous without labels, the accuracy of k-NN classifiers reaches the peak at the middle and then drops.

ble reacts to samples with different difficulties. Interestingly, the maximum accuracy and confidence of ambiguous samples without labels reach a peak at the fifth layer, where a better classification can be found. The downward trend of samples that look like a different class indicates the wrong prediction becomes more confident as the image goes deeper. For samples of the other two difficulty forms, the maxima are at the last layer. We then visualize the confidence of the ground truth class of misclassified ambiguous w/o labels samples. It have been shown that the model indeed gradually becomes more and more confident about its incorrect predictions. This observation aligns with several recent literature. The searching process is detailed as follows. To begin, the dataset is split with different random seeds for ensemble training. We tune the number of the ensemble with $[4, 16, 64]$ ResNet18 models trained with two partitioning strategies as $[5:5, 8:2]$ on CIFAR10-H. After some trial and error, we opt to train 64 models using $8:2$ random partitions for CIFAR100. For hyper tuning, we performed 100 epochs of standard i.i.d training with stochastic gradient descent. For both CIFAR10H and CIFAR100 datasets, we choose the 3 most accurate and stable training configurations. Next, we add a learning rate scheduler that reduces the learning rate with a cosine function. We also implement MLP and VGG16 for PD evaluation. Although we do not have enough time to well tune the VGG16 ensemble, we observe a similar result shown in Fig. The results of using 30 and 300 k-NN neighbours are consistent (Figure26;Figure27).



(a) 1st k-NN (b) 3rd k-NN

(c) 4th k-NN (d) 9th k-NN

Figure 25: Deep k-NN becomes more confident on its misclassified predictions as it goes deep

3.2.2 Discussion

Currently, there exist two problems with prediction depth. Firstly, a problem shared by PD and C-score is how to find proper random splits for a new dataset. On the one hand, if the test split is too large, the model will suffer from overfitting, and predictive error will be large. This leads to inconsistent predictions across the layers and PDs are large for most of the samples. On the other hand, if the train split is large, it takes more models to evaluate validation time PD. We use 5 : 5 training and validation splits as a comparison. Figures have shown 8 : 2 data splitting works better than 5 : 5. The gap between training and testing becomes smaller which shows less overfitting. In the 2D histogram, more samples move to the upper right region as ambiguous. For CIFAR10-H, patterns of difficulty reduce to two. Sharp and narrow spikes appear in the histogram which causes trouble for training data scheduling.

Searching for k nearest neighbours in a big support set can be an engineering problem in terms of time and memory. With a large feature bank, computing the similarity matrix can cause memory overflows when operations include memory copying or garbage collection are not carried out swiftly. For instance, if we use the entire CIFAR100 as the support set, the feature bank of the first three layers takes up 4.9 gigabytes of memory. An efficient way is to implement k-NN classification with Facebook AI Similarity Search (FAISS) library. Under the hood, it uses multi-threading to perform parallel searches on multiple GPUs and BLAS libraries to optimize matrix multiplication.

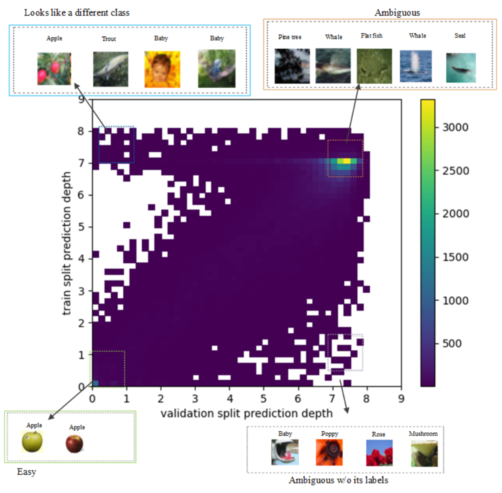


Figure 26: Overconfidence would allocate a weight in the greatest resemblance to the most adjacent k-NN neighbour and wipe out the results from other neighbours. For easy samples, this means all examples are given the same PD values as "easy". By contrast, for ambiguous samples, inconsistent predictions will appear across layers, and samples will be seen as equally "hard".

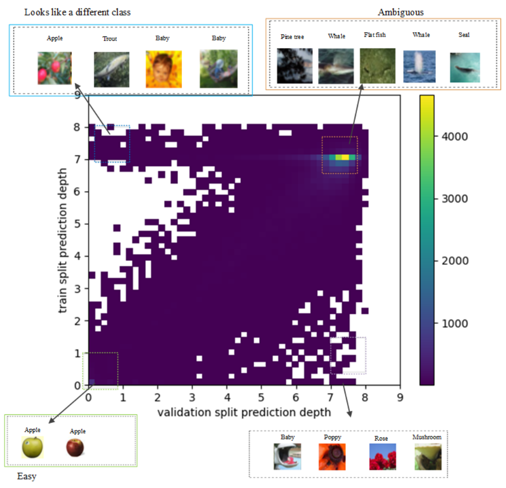


Figure 27: confidence of knn3

Secondly, deep k-NN classification relies heavily on the natural clustering effects of deep neural network and model calibration. Inconsistency can present when model is overconfident or underconfident. This is also probably because no explicit constraints are given on the intermediate latent spaces. Even though cosine distance removes the magnitude of representations, hidden features may not preserve the same angular commonality due to the strong non-linearity posed by the neural network.

Given a certain amount of training data, one can expect prediction depth to be "reliable enough" when a minimum classification rate is reached. For a larger scale dataset with many classes, this mean a larger train split and more runs.

Finally, the blue curves indicate the intermediate layer where the model captures better knowledge of ambiguous samples without labels than other layers. This opportunity motivates us to develop difficulty aware knowledge distillation in our future work.

3.3 Angular output magin

Motivated by two types of difficulty offered by human categorization, we hypothesize an ideal difficulty measurer should take both inter-class and intra-class variance into account. Inter-class variance maximization directly relates to classification by maximizing between-class distance. By contrast, intra-class variance reflects within-class distance and can affect difficulty measurement in a less observable way. When objects are photoed with low resolutions or partially occluded, annotators can possibly infer the labels with background information according to their knowledge. Let us make this more specific with an example. Dogs and deers are more regular grassland visitors than cats. Being on grass may encourage human annotators to inference and vote for these two animals rather than cats when occlusion occurs. Consequently, with the same amount of difficulty from objects, a grassland background helps us more with labelling a deer than a pile of white background. Intra-class variance can also contain ambiguous content that resemble features of another class. Therefore, it is probably safe to conclude that intra-class variance also plays an important role in inferencing the right class.

3.3.1 Inter-class variance and intra-class variance

Here, we build a toy binary classification example to better illustrate this issue and show its connection to representation learning. In Figure, data of two classes are generated from two clusters. As discussed in section 2, an ideal difficulty metric should be precise and generalize to downstream tasks. Specifically, the metric should reflects sample regularity and hardness w.r.t classification. In this case, the first term can be measured by distance or similarity with ground truth class representations. The second term can be calculated by distance to the decision boundary.

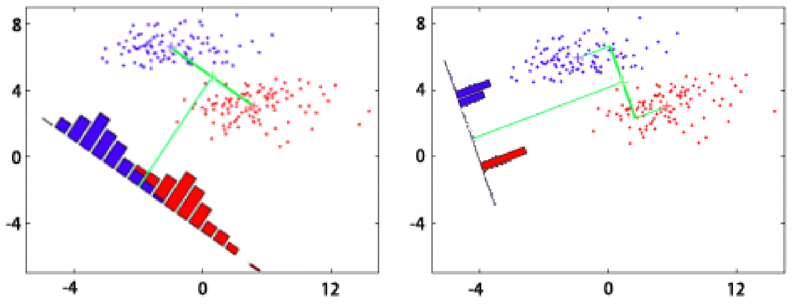


Figure 28: Balancing the right amount of variance

We use two counter examples to highlight the importance of learning a "correct enough" latent space from data with Figure 28. We use neural networks with ReLU activation functions to project input points so the projection is nonlinear. In the first counter example, we use the an objective function of that is similar to LDA to optimize the projection. That is to maximize between-class variance and minimize within-class variance. This leads to a latent space where samples of the same class are projected into a virtually tiny region as shown in the right subplot. Although centres of two classes can be easily estimated, samples are almost equally close to their centroids and far away from the decision boundary. Their difference in terms of difficulty becomes indistinguishable. In the second example, as shown in the left subplot, the intra-class variance is too large and inter-class variance is not large enough. Data points are scattered loosely in the projected plane. As a result, means are not very representative as class representations for members of that category. Additionally, there are always samples of the same class falling on different sides of any potential decision boundary. It is therefore improper to use distance to the decision boundary as difficulty score as the projected plane is improper for classification.

These two counter examples highlights balancing the right amount of inter-class variance and intra-class variance is a important factor to difficulty measurement. So in this section, our goal is to find such a latent space where these two classes of variance easy to compute and well-balanced.

3.3.2 Find a space with deep metric learning

Deep metric learning (DML) methods provide paradigms for this purpose by learning a space where similarity can be easily measured and compared. These approaches have shown promising results in image classification and have been widely used in zero-shot learning, face verification and person re-id. A large number of methods has studied the image classification problem on the latent space with all kinds of distance metrics, but the problem of measuring difficulty on the latent space has not been fully investigated.

In image classification, data are high dimensional and a shared problem of these

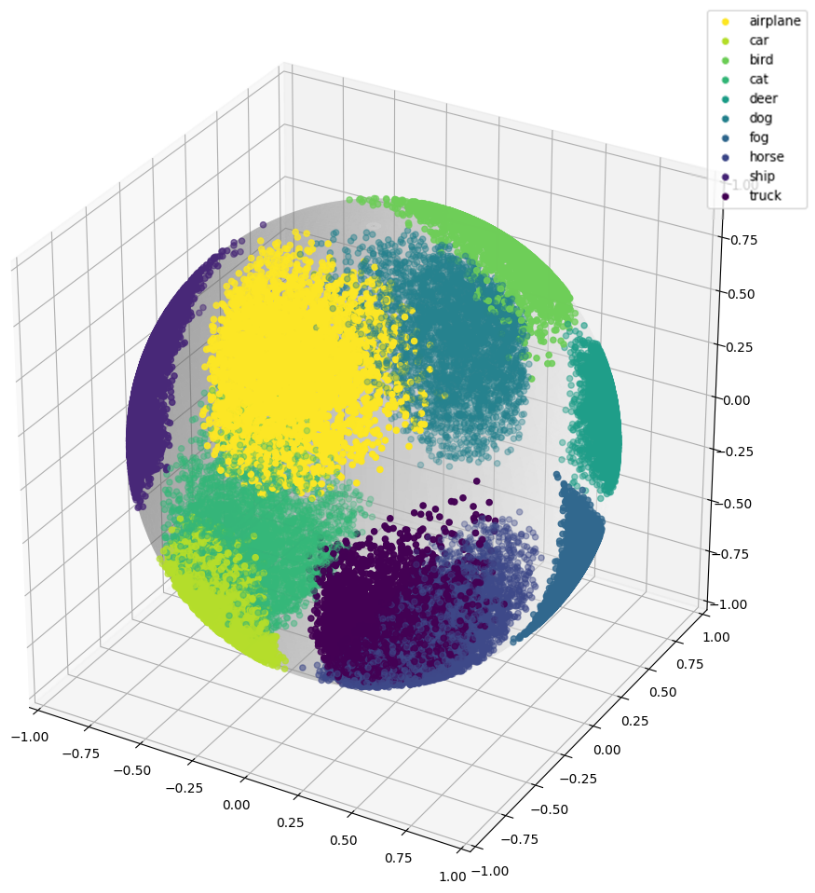


Figure 29: Image representations are projected to an angular spaces

Euclidean methods is that the means estimated in the latent space are sensitive to noise. As an alternative, cosine distance removes the magnitudes of the two high dimensional representations and measures the similarity with their angle. After normalization, image representations are projected on a hyper sphere shown Figure 29. Over the last few years, a group of softmax losses are proposed to maximize discrimination in classification problems. Angular softmax Li et al. (2018) initially normalize the speech representations and the weight parameters of a softmax classifier before computing cross-entropy loss. This allows them to measure class representations during mini-batch training. Following this work, in image classification, Deep spherical embeddings (SphereFace)(Liu et al., 2017), Large margin cosine loss (CosFace)(Wang et al., 2018), and Additive angular margin Loss (ArcFace)(Deng et al., 2019) further reduce intra-class variance by adding a margin on angles or cosine terms.

However, these methods mainly focus on discriminating features of different classes on the latent space. Our goal is to find a better space (hyper ball) where difficulty scores are well-defined (well calibrated) shown in Figure32. In other words, one can find a balance between inter-class representations and intra-class representations on this latent space. Following (Wang et al., 2018), we find a proper difficulty metric by first train a candidate latent space for difficulty measurement with angular (normalized) softmax classifier. Formally, the probability of sample i being in class y_i is modelled by angular softmax classifier as

$$P(y_i|\hat{W};\hat{r}) = f(\hat{W}, g_\phi(\hat{x}_i)) \tag{3.1}$$

$$\hat{W} = \frac{W}{\|W\|} \tag{3.2}$$

$$\hat{r}_i = \frac{r_i}{\|r_i\|} \tag{3.3}$$

$$r_i = g_\phi(\hat{x}_i) \tag{3.4}$$

where $r_i \in R^d$ are the angular representation of the input sample xi; r_i is the nonlinear projection provided by a parametric feature encoder g_ϕ. $W \in R^{Cxd}$ is the weight of the final linear layer of the softmax classifier. Given normalized representations vectors, the cosine can be easily computed by the inner product of w_k and r_i. We can jointly optimize ϕ and W by minimizing the negative log likelihood as

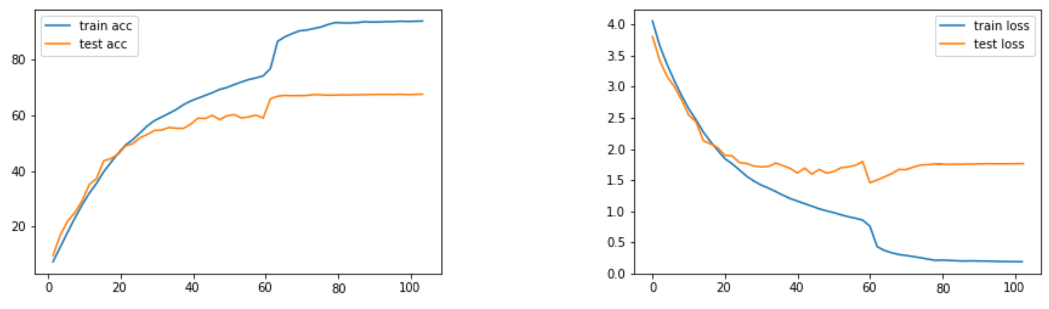
$$L = \frac{1}{N}\sum_{1}^{N} -\log \frac{exp((W_{y_i} \cdot r_i)/t)}{\sum_k exp(W_{y_k} \cdot r_k)/t} \tag{3.5}$$

$$cos\theta(W_{y_i}, r_i) = W_{y_i}^T r_i \tag{3.6}$$

where t is the temperature which is widely used in contrastive learning representation learning, θ is the angle between the kth row vector w_k of weight W and angular representation r_i.

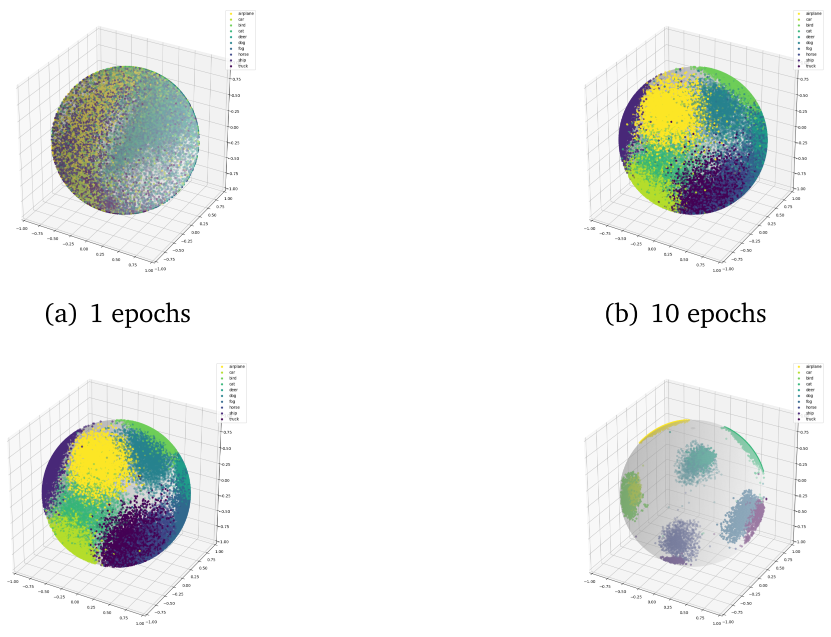
We train the feature encoder on CIFAR10-H for 100 epochs and test the model on CIFAR10 training set. We get the following training dynamics in Figure 30. Note that CIFAR10-H only have 10,000 samples.

Like training with cross-entropy loss, softmax classifier can get overconfident and



(a) Accuracy (b) Loss

Figure 30: Training dynamics of normalized softmax classifier



(a) 1 epochs (b) 10 epochs

(c) 60 epcohs (d) 100 epochs

Figure 31: Visualization of angular space

the generalization gap becomes obvious after 60 epochs. Even when the model is trained with label smoothing, it overfits the soft labels. The model possibly begins to memorize ambiguous samples and learn a complex function. For representation learning, a traditional way to prevent overfitting is to apply early stopping with a k-NN monitor on a hold-out set(He et al., 2020). This can be complex for a small dataset, we introduce a post processing strategy as alternative to alleviate the influence of overfitting.

To better understand the latent space we learnt, we freeze the parameters of the feature extractor and finetune two FC layers on top. This visualization technique quite popular in recent studies (Müller et al., 2019). Figure 31 shows the positions of CIFAR10-H examples on the latent space.

For Large margin softmax loss, when the cosine distances are smaller than the margin, the function values are pushed towards the saturation area of the softmax function. This results in weaker backpropagate gradients and hence a slower convergence than NSL. Notably, if the margin is initially set too large and the batch size is small, the gradients can be dominated by a few data points and the update directions can be noisy. This issue leads to an uneven angular space.

3.3.3 Which angle for difficulty

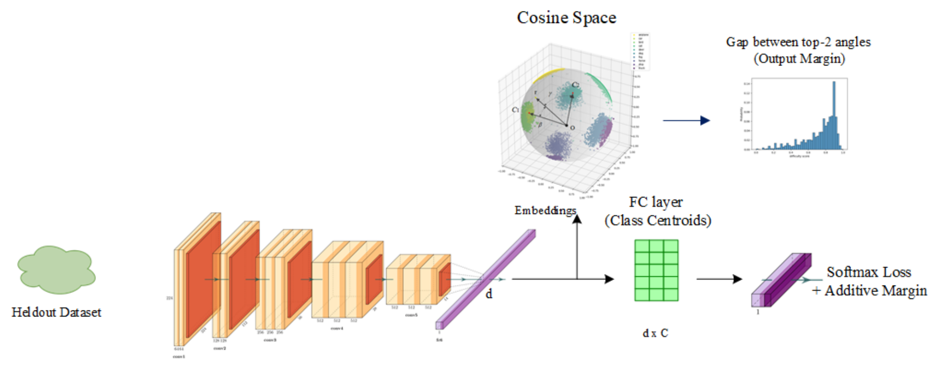

Cosine Space
Gap between top-2 angles (Output Margin)
Embeddings
FC layer (Class Centroids)
Heldout Dataset
d
d x C
Softmax Loss + Additive Margin

Figure 32: Compute angular output margins with the gap between top2 cosine distance

Since we have modeled the probability with scaled cosine values. It is natural to measure difficulty scores as the angle between example representation and its class representation. We explore how to estimate class representations and conclude three different ways. The traditional way is to compute the mean of sample of the current class. Although this method is straightforward, it is sensitive to noise. Another way is to estimate class representations with a support set compose of pristine examples. This method borrow the idea of (Patrini et al., 2017) where they measure the noise transition matrix with pristine class example to tackle label noise. Since we have human scores at hand, we use the top 10 examples as our support set. The refined class representations are shown in Figure 33. The third way is to estimate $\hat{W} \in R^{Cxd}$ as class angular representations because $\hat{W}$ exactly has C vectors as the number of classes. If we directly use W, it will be very noisy due to mini-batch training so we keep a buffer for W and update it with momentum. This is similar with the update rule of the memory bankHe et al. (2020). Here we use the angular gap instead of the angle between example representation and its class representation because the latter is exactly the output logit. The angular gap is similar with the output margin

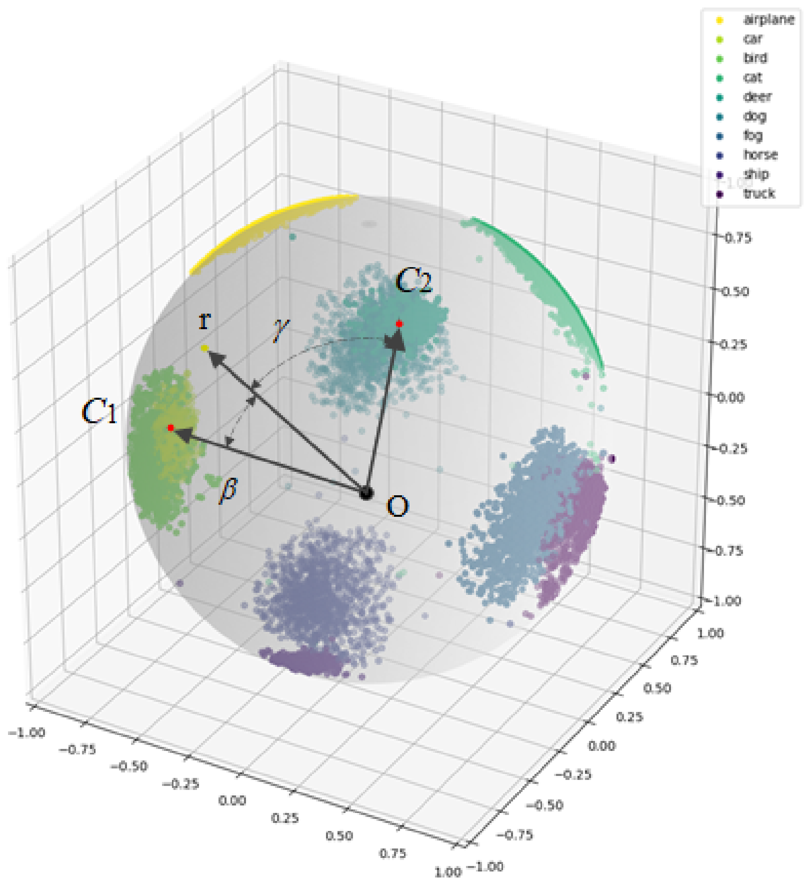
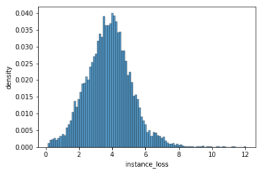
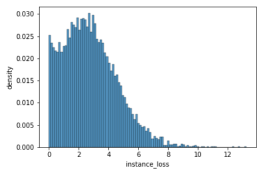


Figure 33: Angular gap is the difference between the cosines of top2 predictions

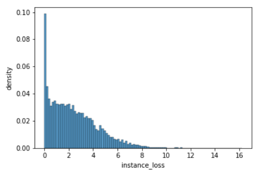
discussed in section 2. We visualize the distribution of normalized angular gap[2] in Figure 35.



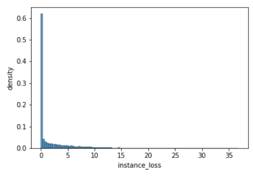
(a) 1 epochs



(b) 10 epochs



(c) 60 epcohs



(d) 100 epochs

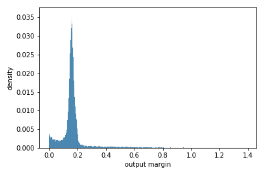
Figure 34: The distributions of softmax classifier's loss

3.4 Comparing different metrics on CIFAR10-H

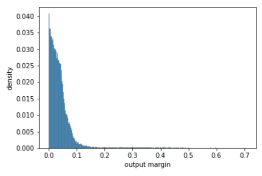
Finally, we reach the end of this section by comparing different difficulty scores on CIFAR10-H with a correlation matrix (Figure 36). For human scores, we have cross-entropy loss per image, output margin and reaction time. For the score based on Monte Carlo sampling, we have C-score offered by Jiang et al. (2020). For the score based on training dynamics, we have forgetting events. For pretrained empirical loss, we have ResNet18 output margin and ResNet18 loss. For the two dimensional difficulty score, we have prediction depth. For score based on DML,we have angular gap (output margin). We compute Spearman rank correlation in a pairwise way. And we show the correlation matrix in Figure 36. We also report other baseline difficulty scores in Figure37.

Pretrained ResNet18 loss is uncalibrated which follows self-paced learning's convention reported in literature. By contrast, ResNet18 output margin use a hold-out dataset. The uncalibrated ResNet18 model report a entirely uncorrelated order with others. It is interesting to see forgetting events also show less correlation with other scores due to uncalibrated training dynamics. Human loss, C-score and angular gap show more correlation which probably means they belong to the same category, empirical loss. Prediction depth is likely to be an outstanding metric, and it shows an almost even correlation with C-score and forgetting events. This correlation matrix highlights the importance of being unbiased and well-calibrated, the abilities to report a right amount of confidence or ambiguity during difficulty measurement.

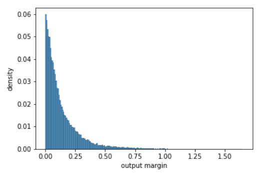
[2]We normalize this quantity because we need the metric to between 0 and 1 for self-paced curriculum learning introduced in Section 5



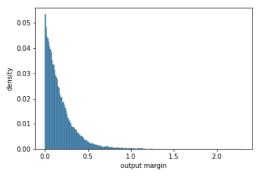
(a) 1 epochs



(b) 10 epochs

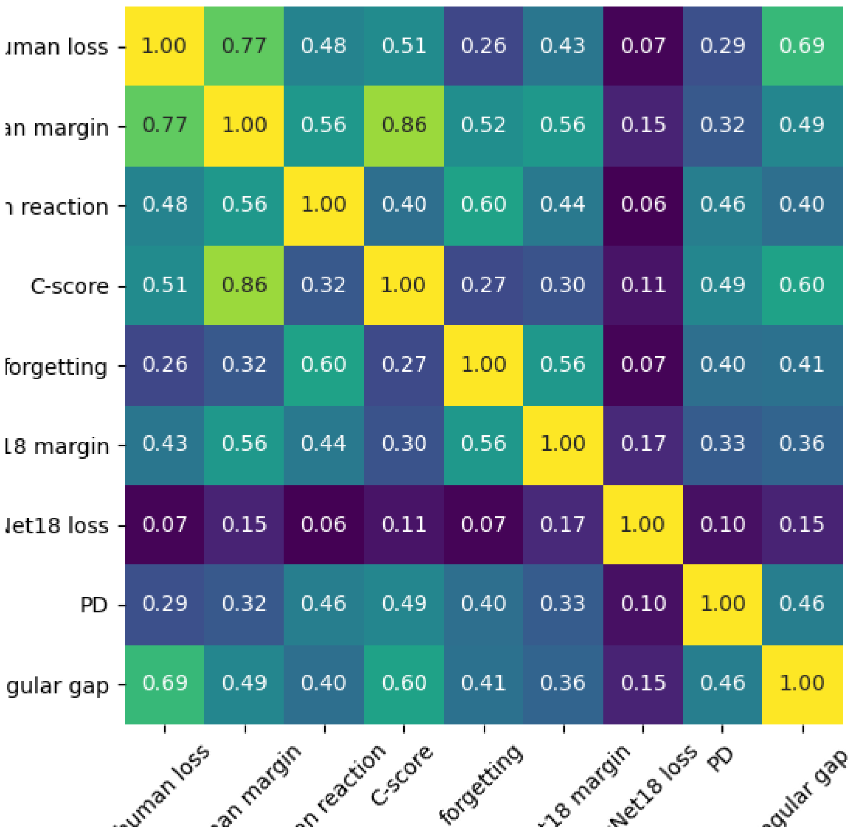


(c) 60 epcohs



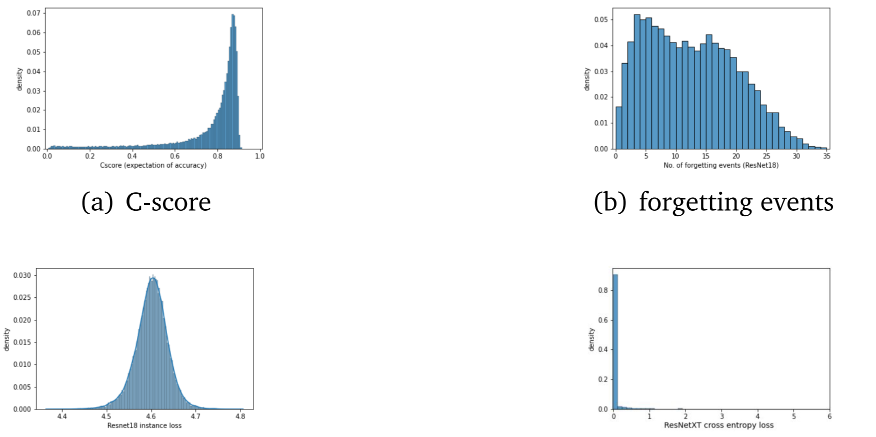
(d) 100 epochs

Figure 35: The distributions of Angular gap



	ıman loss	ın margin	ı reaction	C-score	forgetting	L8 margin	Iet18 loss	PD	gular gap
ıman loss	1.00	0.77	0.48	0.51	0.26	0.43	0.07	0.29	0.69
ın margin	0.77	1.00	0.56	0.86	0.52	0.56	0.15	0.32	0.49
ı reaction	0.48	0.56	1.00	0.40	0.60	0.44	0.06	0.46	0.40
C-score	0.51	0.86	0.32	1.00	0.27	0.30	0.11	0.49	0.60
forgetting	0.26	0.32	0.60	0.27	1.00	0.56	0.07	0.40	0.41
L8 margin	0.43	0.56	0.44	0.30	0.56	1.00	0.17	0.33	0.36
Iet18 loss	0.07	0.15	0.06	0.11	0.07	0.17	1.00	0.10	0.15
PD	0.29	0.32	0.46	0.49	0.40	0.33	0.10	1.00	0.46
gular gap	0.69	0.49	0.40	0.60	0.41	0.36	0.15	0.46	1.00

Figure 36: Correlation matrix of different difficulty scores on CIFAR10-H



(a) C-score (b) forgetting events

(c) Uncalibrate ResNet (d) ResNet Output Margin

Figure 37: The distributions of difficulty metrics

4 Handcrafted curricula

As discussed in Section 2, For handcrafted curricula, designers must first choose a proper difficulty metric according to the downstream task before choosing the right pacing function. This is because pacing functions need orders computed from previous runs. The classic combination has generated numerous curricula learning methods. As a handy tool, continuous schedulers are often problem agnostic and can be implemented with a few lines added upon common data loaders. However, Wu et al. (2020) shows curriculum learning results in no improvement for image classification. In this project, we beg to differ this conclusion by demonstrating curriculum learning helps image classification.

As have been discussed in Section 2, Wu et al. (2020) might violate the easy-to-hard principle. To test this doubt, we use human scores as difficulty scores and linearly add in training examples of CIFAR10-H. This configuration is called standard curriculum and has a fixed order (Bengio et al., 2009). In this project, we run three experiment with different random seeds for each curriculum parameter and report the average accuracy . During each run, we fix the seeds of random, NumPy and PyTorch packages. Then we turn on CUDNN deterministic setting which may probably slow down training. However, this allows us to compare different methods head-to-head. We train ResNet18 models from scratch in these runs equally for 40000 iterations. We apply standard data augmentation of CIFAR10. The optimizer we use is SGD with a momentum of 0.9. The learning rate starts from 0.1 and gradually decrease to 0 with a cosine annealing strategy. We create a search grid for standard curriculum and run our experiments on Slurm. We report the highest accuracy this time in Figure 39.

This causes spikes in the accuracy curves as shown in Figure 41 and Figure42. The results of the same configuration but different random seeds vary a lot and collapse can happen during training (Figure40).

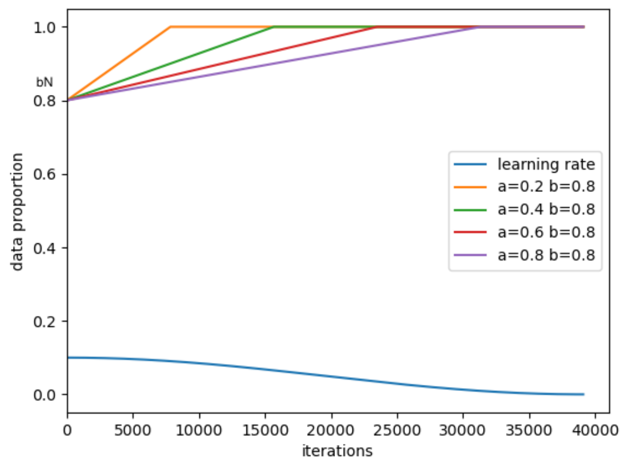


Figure 38: This plot shows four sets of curriculum learning parameters. They all start from 80 percentage of training samples (easy) and gradually add in difficult samples until 20/40/60/80 percentage of total iterations. These pacing functions cooperate with the learning rate scheduler. Together, they control the update gradients from samples of different difficulty levels

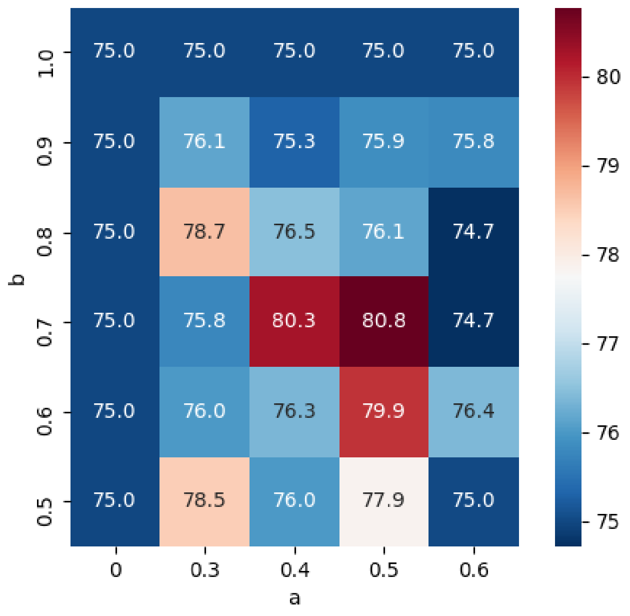
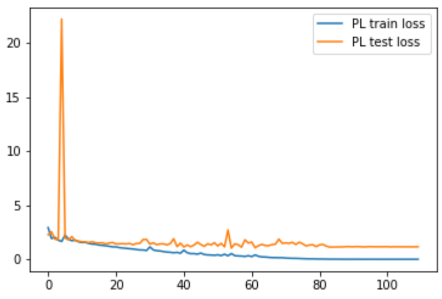
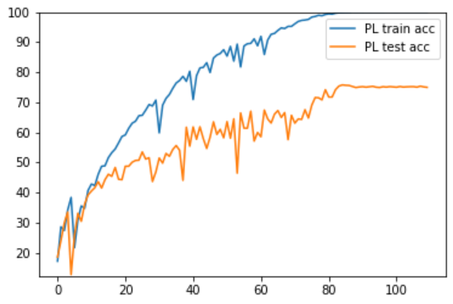


Figure 39: Grid search of standard paced learning on CIFAR10-H

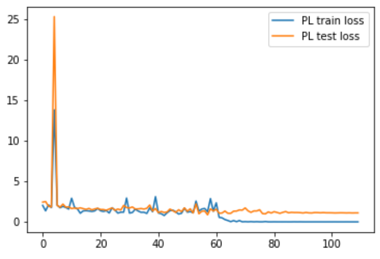


(a) Loss

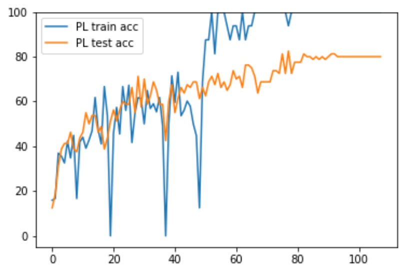


(b) Accuracy

Figure 40: Damaging training with standard PL strategy ($a = 0.5, b = 0.7$)

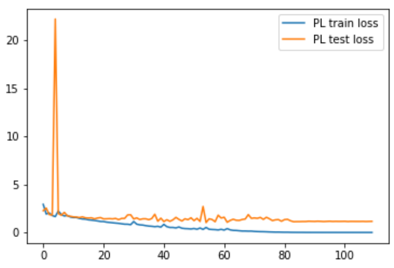


(a) Loss

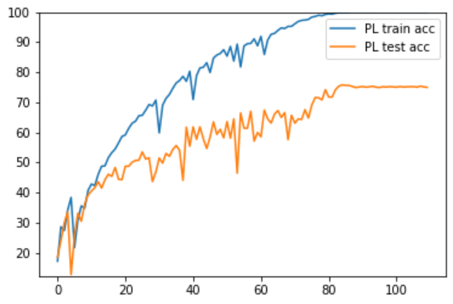


(b) Top1 Accuracy

Figure 41: Improving training with standard PL strategy ($a = 0.5, b = 0.7$)



(a) Loss



(b) Accuracy

Figure 42: Damaging training with standard PL strategy ($a = 0.5, b = 0.7$)

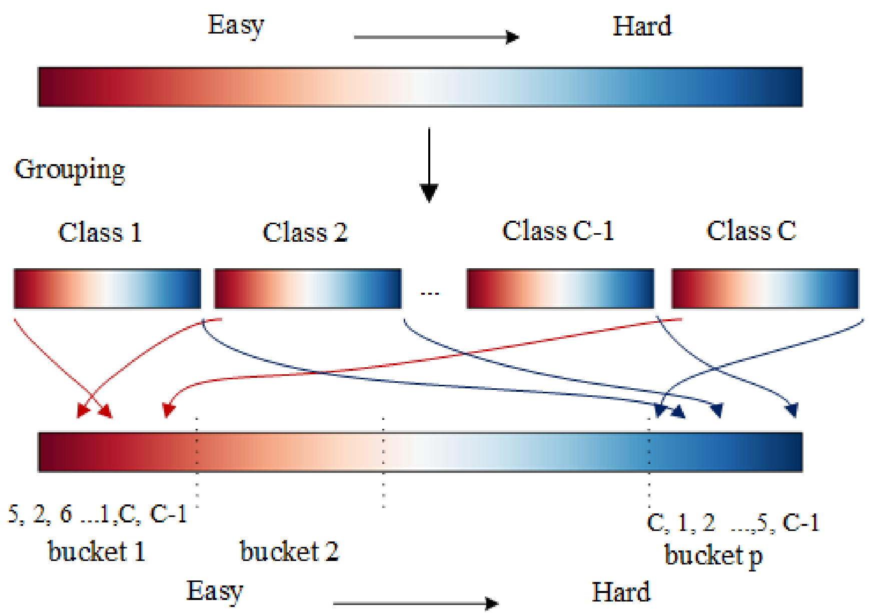


Figure 43: class-balance aware paced learning scheduler

4.1 A class-balance aware scheduler

To tackle this, we propose a class-balance aware paced learning scheduler shown in Figure 43. The idea is to introduce local randomness with a class balancing strategy. The algorithm can be explained as follows, we first group sorted training data with their labels as a dictionary. Then we apply standard continuous paced learning steps that choose the number of add-in samples with a and b. Next, we use the precomputed dictionary to generate indices for class-balanced buckets. The generator first samples a class and then sample a data point of this class from easy to hard. In this case, the size of each bucket equals the number of classes. This class-balancing strategy introduces randomness to the local level but maintains the easy-to-hard data structure at the global level. The details of our continuous paced learning scheduler is shown in Algorithm 2. The improvement over standard i.i.d training is shown in Figure 52.

As shown in Algorithm 3, the scheduler can be a work-out-of-shelf module for discrete paced learning after changing a few lines. The embedded class-balancing strategy not only shuffles the data within the bucket but can handle imbalanced classification by upsampling the minority class during training. As we will show in the empirical experiments, this scheduler makes a tradeoff between i.i.d training and easy-to-hard curriculum. It provides a more stable performance boost. The scheduler works well with most difficulty metrics on image classification tasks and leads to consistent results. Although we only have a handcrafted offline version now, we are confident that this scheduler can be extended to online curriculum learning.

Algorithm 2 Class balance aware paced learning(continuous

```
Input: S:training dataset; M: difficulty measurer; $\lambda(\cdot)$:pacing function; b: initial
   training proportion; a:milestone when all data enter; $g(\cdot)$: bucket generator
Output: $W^*$: optimal model parameters
  $S, order \leftarrow sort(S, M)$;
  $\{l : order\} \leftarrow group(S, order)$                         ▷ group order w.r.t labels
  $S^0 = \lfloor b * S \rfloor$                                      ▷ warm up bucket
  $train(S^0, M)$
  $S^{hard} = S \backslash S^0$
  for epoch = 1,2, ...K do
      $\lambda_t = \lambda(k, a, b); \quad 0 < \lambda_t < 1$                ▷ get current $\lambda$ training data
      $S^t = \lfloor \lambda_t * S^{hard} \rfloor$
      $S^{train} = S^{train} \cup S^t$                            ▷ add in more difficult samples
      $p = S^{train}$ |C                                   ▷ No. of class-balance training buckets
      $[B^1, B^2, \ldots, B^p] \leftarrow \quad g(S^{train}, \{C : order\}, p)$   ▷ get class-balance training buckets
      $order^t \leftarrow [B^1, B^2, ...B^p]$
      $S^{train} = S^{train}[order^t]$
      while not stagnate train for k epochs do
          $train(S^{train}, M)$                           ▷ apply normal mini-batch training
      end while
  end for
  return $W^*$
```

Algorithm 3 Class balance aware paced learning (discrete)

```
Input: S:training dataset; M: difficulty measurer; g:training bucket generator
Output: $W^*$: optimal model parameters
  $S, order \leftarrow sort(S, M)$;
  $[S^1, S^2, ...S^T] = S$
  $S^{train} = \phi$
  $\{l : order\} \leftarrow$ group order w.r.t labels
  for t = 1,2, ... T do
      $S^{train} = S^{train} \cup S^t$                            ▷ add in more difficult samples
      $p = S^{train}$ |C                                   ▷ No. of class-balance training buckets
      $[B^1, B^2, ...B^p] \leftarrow \quad g(S^{train}, \{C : order\}, p)$      ▷ get class-balance training buckets
      $order^t \leftarrow [B^1, B^2, ...B^p]$
      $S^{train} = S^{train}[order^t]$
      while not stagnate train for k epochs do
          $train(S^{train}, M)$                           ▷ apply normal mini-batch training
      end while
  end for
  return $W^*$
```

4.2 Evaluation

In this project, we have trained over 900 models to understand curriculum learning. Here, we opt to show the usefulness of paced learning for image classification. Generally, it is difficult to evaluate curriculum learning methods without exhaustive grid search. For grid search, we run each configuration with three random seeds and report the average accuracy. During each run, we fix the seeds of random, NumPy and PyTorch to a constant. Then we turn on CUDNN deterministic setting which may probably slow down training, but this allows us to compare different methods head-to-head.

We train the baseline with random i.i.d sampling on both CIFAR10-H[3] and CIFAR100. The batch size we use for both datasets is 128 and the number of total training iterations is 20000. We use the standard data augmentation technique for CIFAR datasets that include random crop with padding, random horizontal flip and normalization. For optimization, we use a momentum stochastic gradient descent (SGD) optimizer with momentum equals 0.9 and the weight decay parameter equals $5e-4$. We use a cosine annealing strategy that starts from 0.1 and ends with 0 because difficult samples are added gradually and frequently in this project. We choose not to use Dropout or advanced data augmentations such as Cutout to ensure consistent results. We use linear pacing functions and grid search curriculum parameters a and b. The pacing functions start with b percentage of data and gradually add in more difficult data until aT iterations and then keeps on training towards the end.This configuration is adopted for most of our experiments if the difference is not specifically mentioned.

To highlight the importance of pacing, we perform curriculum learning without

Table 2: Performance of fixed curricula w/o pacing

	CIFAR10-H		CIFAR100	
	Top1 Err.	Top5 Err.	Top1 Err.	Top5
Standard i.i.d[4]	**24.00**	**1.844**	**23.90**	**6.87**
Forgetting events	25.70	1.98	25.68	8.22
C-Score	**24.54**	1.96	**24.95**	**7.33**
Human score	25.03	**1.92**		

pacing (fixed curriculum) with different forms of difficulty. To apply this, one sorts samples from easy to hard as a fixed curriculum. The target model learns on this curriculum without changing the sample order. The strategy is similar to teaching a student to recite a passage. We compare their results with standard i.i.d. training in Table2. The curriculum learning w/o pacing function methods neither respects

[3]As discussed in Section 2, CIFAR10-H contains more ambiguous images with human scores.

global difficulty structure nor introduces any randomness during training. The learning the model can learn all data at any time. This results in a lower accuracy than the standard i.i.d. baseline on average.

To compare different pacing functions, we test three pacing functions with our scheduler on CIFAR100. We create three search grids for linear, root and quadratic pacing functions. Figure 44, Figure46 and Figure45 show a similar trend. The upper left region shows higher accuracy while the lower right shows lower accuracy. Figure 44 shows the optimum in (0.2, 0.8). This means the model performs better when the curriculum starts with about 80 per cent of data and ends adding data at 20 per cent of the total iterations. The lower right region also proves the importance of assigning enough informative materials within the critical period **??**. The experiments with root pacing functions show fewer different results due to faster data adding. Quadratic pacing functions show similar results in the upper left region but visibly worse performance in the lower right region. This is probably because the curriculum has taught the model with too many easy samples and miss the critical period to learn high-frequency features. A widely accepted hypothesis is that the parameters of the network are updated too far away towards the distribution of easy data and can never be pulled back again. Although we did not fix the learning rate, Toneva et al. (2019) test the comparing experiments with fixed learning rate and show similar results. This supports the notion that critical periods cannot be explained solely in terms of the loss landscape of the optimization. Since different pacing functions show consistent results, we opt to use linear pacing functions for the rest of the experiments in this chapter. Another benefit of using linear functions is the interplay between difficulty and optimization is relatively straightforward.
Additionally, we did an ablation study with random curriculum learning, standard PL, PL with our class-balance-aware scheduler and the baseline. Here random curriculum shuffle the precomputed order and use this order to do paced learning. This is the ”standard” method reported in Wu et al. (2020). After grid search, we report the best performed model (Figure53). Our ablation study shows standard PL learns the fastest and achieve the highest test accuracy, but its curves oscillate a lot. Curriculum learning with a random order shows the worst result. Our novel scheduler shows an intermediate test accuracy (about 2% improvement) and also stable performance during training. Additionally, we compare paced learning with different difficulty scores 3. Let us revisit the results of difficulty scores on CIFAR10-H (Figure37;Figure36). Comparing the histograms, we observe smaller gaps between bins in C-score than forgetting events. This indicates C-score approximates a smoother function than forgetting events. Smoothness allows us to tame curriculum learning with a fine-grained order. Otherwise, schedulers sort samples with equal scores randomly. By contrast, the histograms of human annotations are spiky or uneven but training with human scores shows better performance than training with forgetting events (Figure47). This is probably because humans are better at marking ambiguous samples than neural networks. Humans can vote out an almost even result, e.g., like 65:35, on ambiguous samples. Whereas uncalibrated neural networks may all correctly classify those samples and output similar difficulty scores as easy samples.

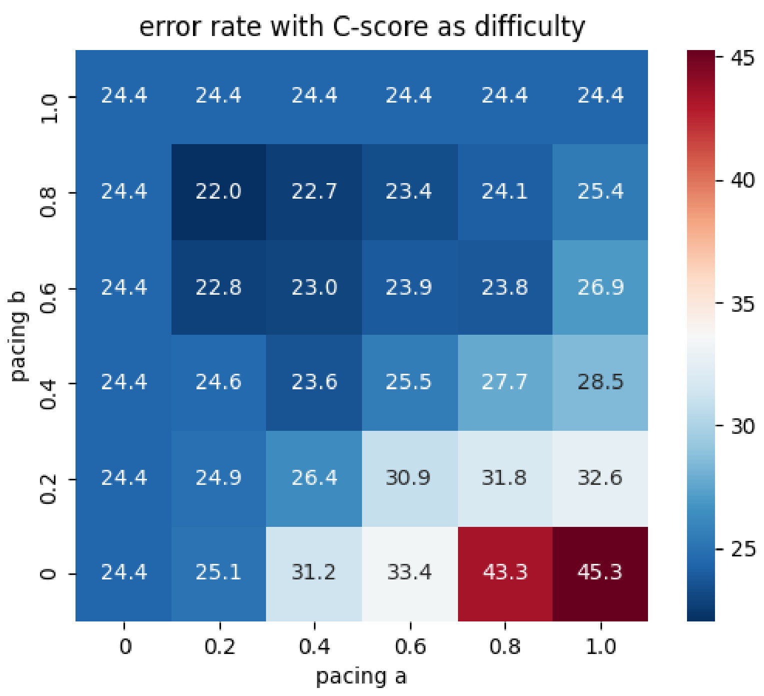


Figure 44: Paced learning with linear pacing functions

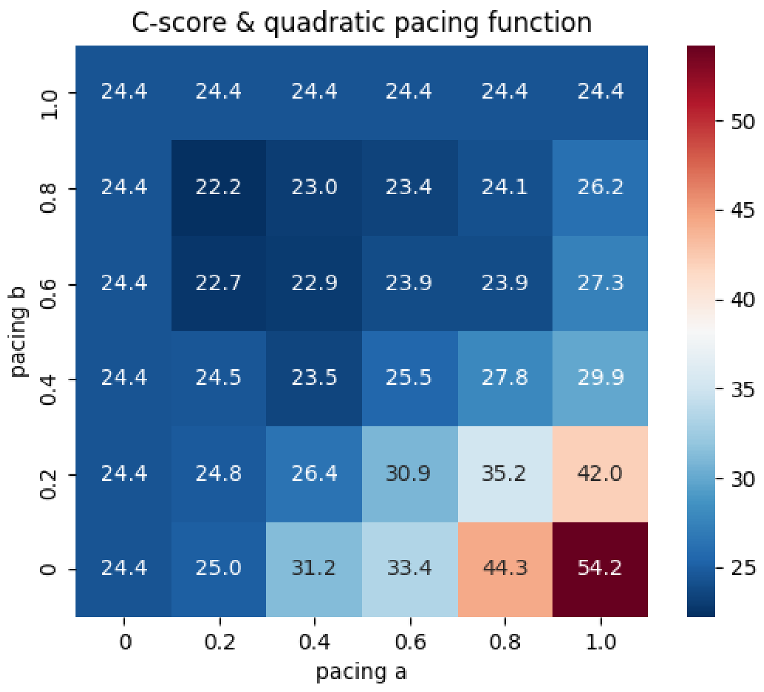


Figure 45: Paced learning with quadratic pacing functions

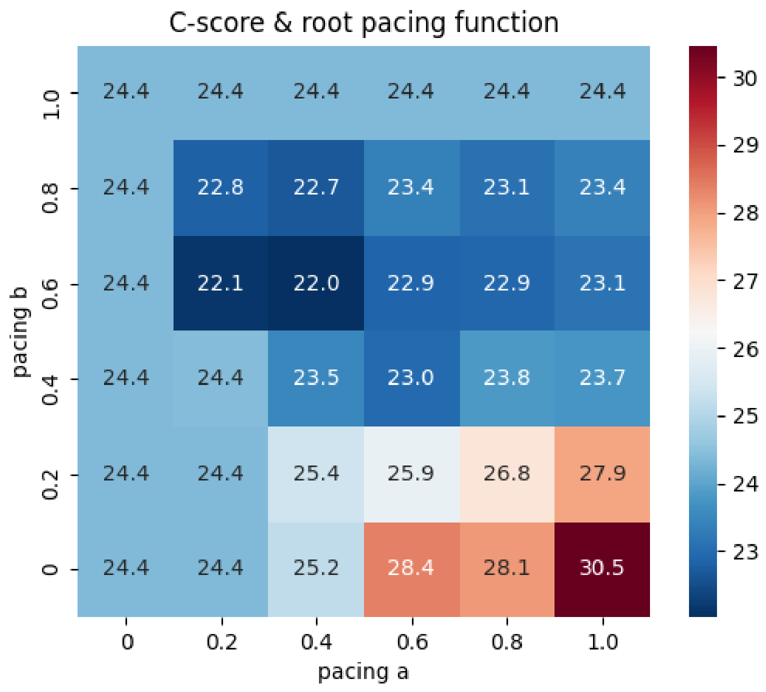


Figure 46: Paced learning with root pacing functions

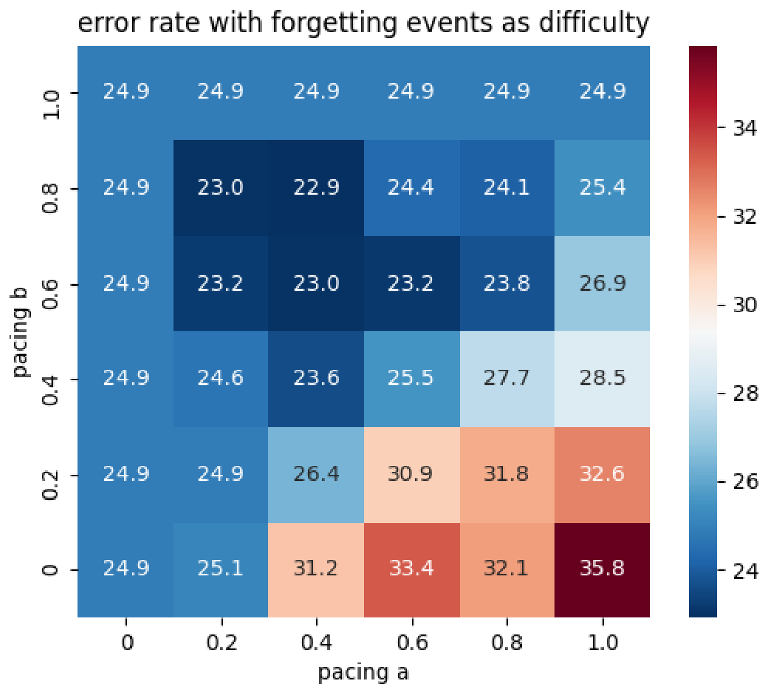


Figure 47: Paced learning with forgetting events as difficulty scores(linear pacing functions)

Table 3: Performance of PL on CIFAR10-H and CIFAR100

	CIFAR10-H		CIFAR100	
	Top1 Err.	Top5 Err.	Top1 Err.	Top5
Standard i.i.d	**24.00**	**1.844**	**23.90**	**6.87**
Human Score	21.90	1.51		
Forgetting events	22.57	1.86	22.91	7.68
C-Score	**21.85**	**1.33**	**22.02**	**5.96**
Prediction Depth	**21.74**	**1.21**	**21.86**	**5.90**
Angular Output Margin	22.07	1.50	22.50	6.33

Model calibration solves this issue and contributes to better curriculum learning with C-score or PD than those with human scores on image classification. For C-score, we calibrate every single model in the ensemble. And for PD, we not only calibrate every single model but also every k-NN classifier. There remains an issue to apply calibration techniques and curriculum learning to other imaging applications such as segmentation in our future work.

Furthermore, we compare training with curriculum and anti-curriculum. Our

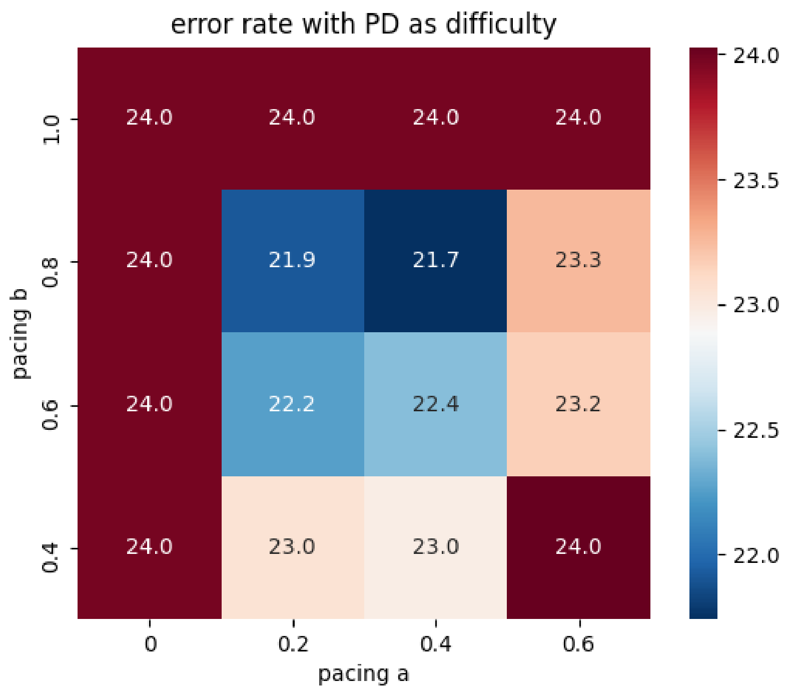


Figure 48: Curriculum learning with prediction depth

results show curriculum learning systematically outperforms anti-curriculum learning on CIFAR100 image classification tasks. This result is different from Wu et al.

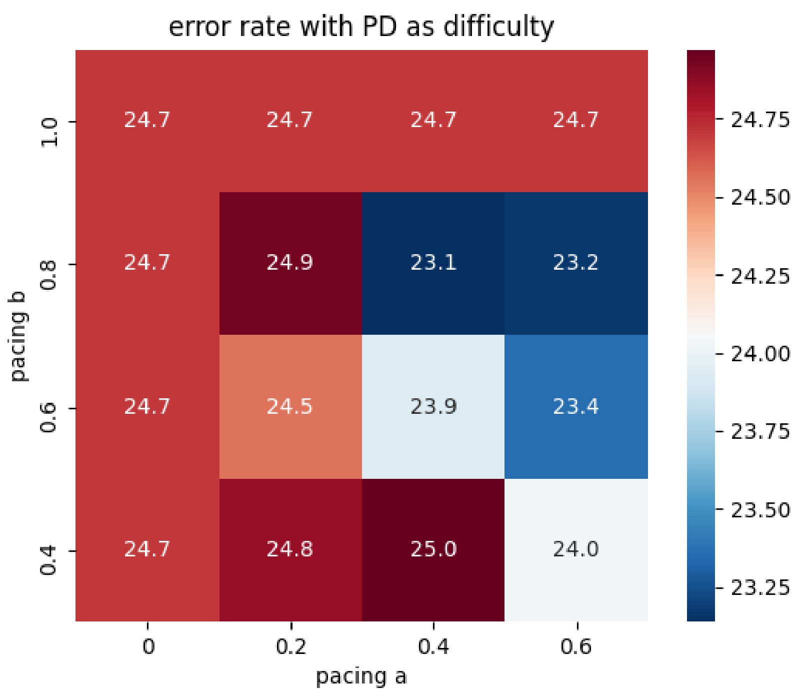


Figure 49: Anti-curriculum learning with prediction depth

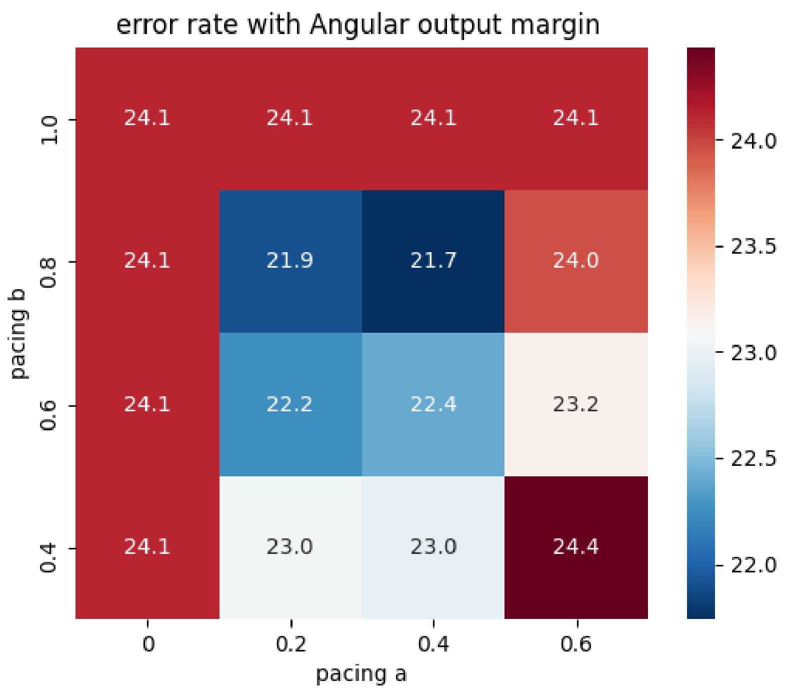


Figure 50: Curriculum learning with angular output margin

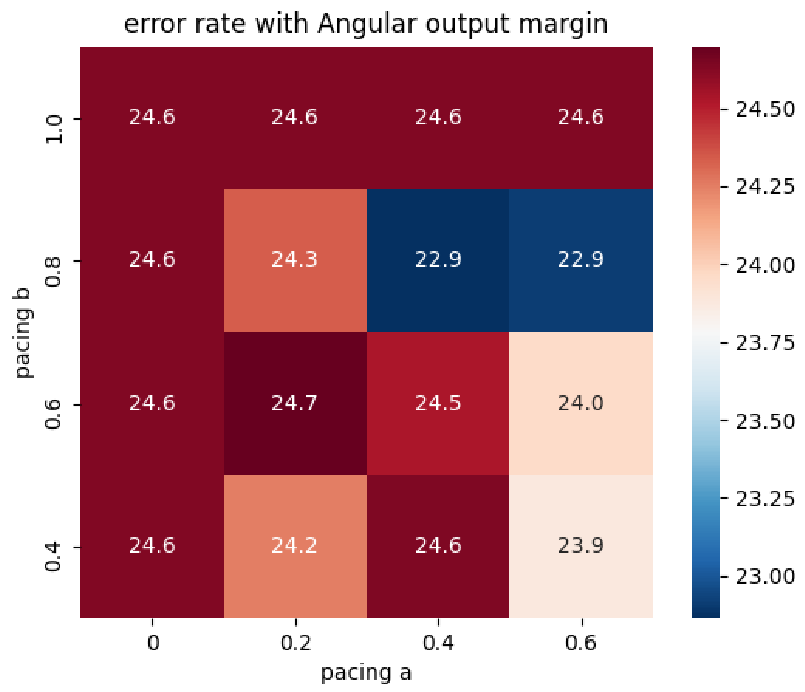
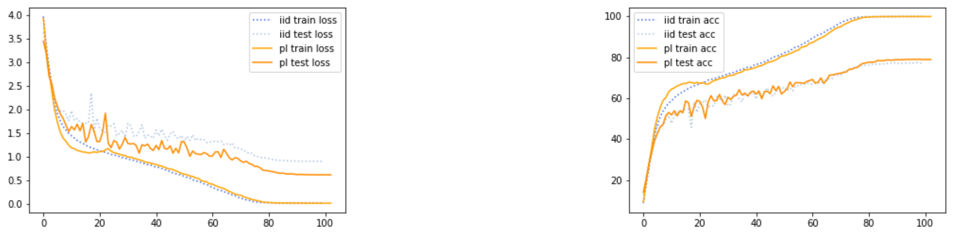


Figure 51: Anti-curriculum learning with angular output margin

(2020). As we point out earlier, our scheduler respects global difficulty structure and follow the principle of curriculum learning right from the initial subset.

Interestingly, an overshoot exists in the early stage of our paced learning with C-

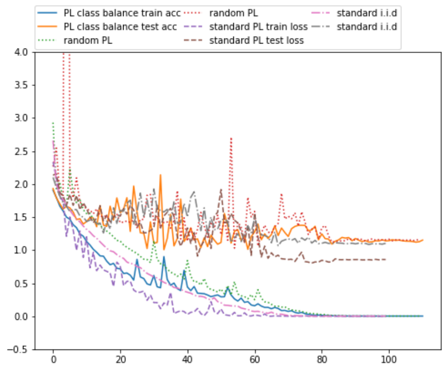


(a) Loss (b) Accuracy

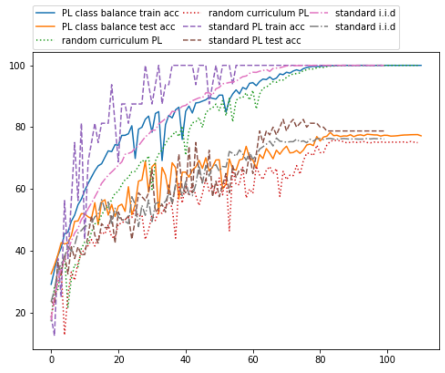
Figure 52: Orange line: Paced learning trains student network on CIFAR100 from easy to difficult with our class-balance-aware scheduler (example difficulty is evaluated with pretrained teacher networks). Blue line: standard random i.i.d training

score. In Figure 52, both training and testing loss decreases drastically at the beginning as the model learns from easy samples. Then we add in hard samples and the training loss levels for several epochs before decreasing to zero. Similarly, accuracy curves show overshooting as well and the test accuracy rises to over 78 after convergence. The existence of overshooting implies the simple function that the model learns cannot predict the hard training samples. This is connected to a recent paper **?** where they observe early layers generalize while later layers memorize. Different experimental paradigms also report DNN learning easy and simple functions first. A well-known benefit of simple functions is they generalize very well. Our method encourages learning easy functions by presenting easy samples first which can be seen from the relatively small gap between training and testing. This phenomenon is also supported by paced learning with forgetting events shown in Figure. To test the degree of generalization gap minimization, we test sharpness-aware minimization (SAM) with the same experimental setup. We use David Samuel's PyTorch implementation and discard label smoothing and dropout to get a fair comparison. We search the neighbourhood size ρ as [0.01, 0.05, 0.1, 0.2] and runs three experiments with each configuration. For the best configuration, we get an average accuracy of 77.90. Although SAM gets marginally lower result, its efficiency is obvious as the algorithm explicitly minimize local sharpness and leads to a "flat" minimum. Whereas paced learning implicitly achieves generalization gap minimization with a manually designed curriculum. This requires considerable hyperparameter searching which can be infeasible for large scale problems.

We also implement and investigate the behaviour of the Baby step scheduler, as a



(a) Loss



(b) Accuracy

Figure 53: Ablation study on CIFAR10-H

standard discrete scheduler. Unfortunately, to implement this scheduler, one needs to carefully consider the number of buckets, bucket shuffling and the number of inner loops. Additionally, one also needs to design a and b for this discrete pacing function. This leads to a considerable searching space. In this project, we manually design these curricula on CIFAR100 with C-score. We performs hyper tuning on the scheduler with the following search grid (Table 4). For simplicity, we shuffle the data within each bucket but keep the order of data buckets. Note that, when a equals 0.0 or b equals 1.0, the algorithm is still different from standard i.i.d training as the global difficulty is preserved. We report the highest accuracy of the best perform model as 0.785. We observe about 2% improvement on CIFAR100.

4.3 Handcrafted curricula for Transfer learning

As pointed out by Bengio, we can take curriculum learning as a special form of transfer learning where the subtasks gradually guide learners towards a better per-

Table 4: Search grid for Baby step scheduler

Variable name	Range
No. of buckets	$[20, 50]$
No. of inner loops	$[1, 2, 3]$
a	$[0.0, 0.3, 0.6]$
b	$[0.6, 0.8, 1.0]$

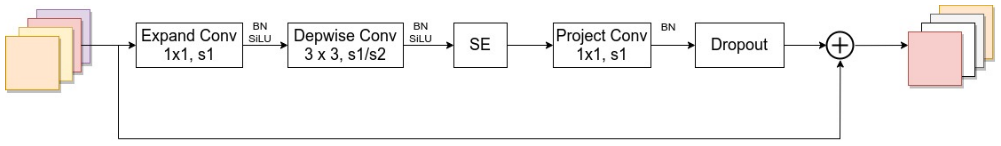


Figure 54: MBConv Block

formance on the final target task. We have also shown presenting samples in a meaningful order improves generalization. These motivate us to test paced learning in the transfer learning setting. In image classification, SOTA convolutional neural networks are designed to be compact and efficient which facilitates deployment on portable devices. However, finetuning these architectures can be a disaster. With fewer parameters, it is a challenging task to transfer the knowledge learnt from the original domain to the target domain. To the best of our knowledge, EfficientV2 is the SOTA model on ImageNet at the moment. In this project, we test the effect of PL on finetuning parameters of EfficientNetV2-B0 learnt on ImageNet to CIFAR100 (Figure 56).

EfficientNet is the SOTA architecture in terms of both accuracy and efficiency on ImageNet. Google Research finds this series of architectures with compound model scaling and neural architecture search **??**. The base block of EfficientNetV1 is the invertible residual block, also known as the MBConv block. Let us take a closer look at this structure and see how it is modified as Fused-MBConv used in EfficientV2. As shown in Fig, the main difference between the MBConv block (Figure 54 and residual block is that the original 2d convolution is replaced with depthwise convo-

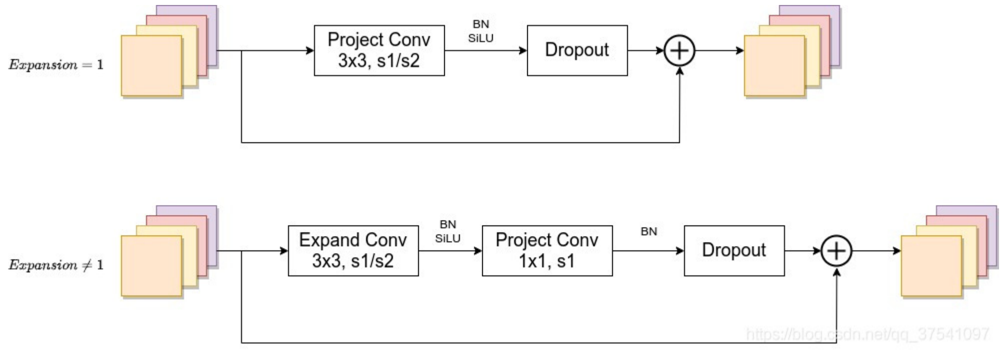
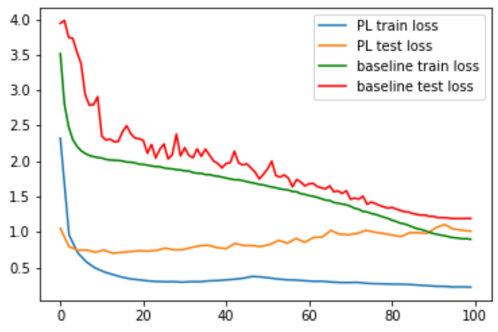


Figure 55: Fused MBConv

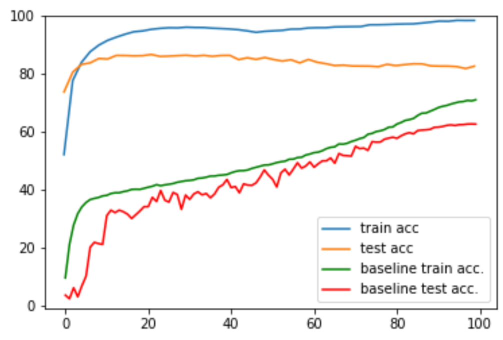
lution followed by a squeeze and excite (SE) block. The depthwise convolution use M filters to separately convolute M feature maps. For the same number of feature maps, this operation is faster than the original so one can add more channels to capture more features. The SE block applies channel attention to those feature maps with its squeeze operation. Finally, the output of the main branch is convoluted to the same size as the input and a dropout is applied. Different from Resblock's wide-narrow-wide structure, MBConv uses a narrow-wide-narrow layout. This allows deep models to learn more features with fewer parameters.

However, an existing problem of EfficientV1 is that using depthwise convolutions in the shallow layers slow down the computation. Although the depthwise convolution has fewer parameters and smaller FLOPs than the original convolution, it is often not possible for this operation to make full use of existing hardware accelerators. In other words, the actual speed is not as fast as expected even though the theoretical complexity is small. Recent research proposes Fused-MBConv to make better use of mobile or server accelerators. The Fused-MBConv block (Figure 55) replaces the main branch of the original MBConv (depthwise convolution and SE block) with a common 2d convolution, as shown in Figure. Additionally, EfficientNetV0 uses drop connect as regularization. Unlike dropout, drop connect randomly discard the inputs of neurons instead of randomly cutting off their outputs. By contrast, EfficientNetV2 uses Stochastic depth as its dropout layer which can adjust the depth by removing entirely removing the main branch.

With variants of MBConv, the EfficientNet models a complex function with limited parameters distributed in a deep architecture. The features across different layers probably need to be precise enough and coordinate like a close-knit. This is likely the reason why finetuning is difficult on these models. We compare finetuning the pretrained EfficientNetV2-B0 on CIFAR100 with paced learning and without. For augmentation, we opt to resize the input image to 224 and apply random horizontal flip but disable cutout for a fair comparison. The batch size we use is 128 and the weight decay is turned off. The initial learning rate is 0.001. Since we look for both convergence rate and accuracy, we set the number of training epochs as 100. For paced learning parameters we search a from [0.1, 0.2, 0.3] and b from [0.6, 0.7, 0.8]. We run three experiments with each configuration and takes an average from them. We also train a baseline with standard i.i.d. We report the best performed paced learning and the baseline in Figure. We made an attempt to use curriculum learning in noisy labelled image classification. We add symmetric label noise to CIFAR10 by splitting the training dataset into several shards and shuffle the labels of 40% data in each shard. To apply curriculum learning with prediction depth, we first train 12 ResNet18 models with different data splits as different pretained teacher networks. Next, we place 9 k-NN probes in each ResNet18. Using these probes, we evaluate both train time and test time prediction depth. We traverse the PD histogram from left to right and bottom to top to get easy-to-difficult order. We use this order to train the student model from easy to difficult with our scheduler and get the final performance as Figure 57. The initial training data is clean ($b = 0.5$),

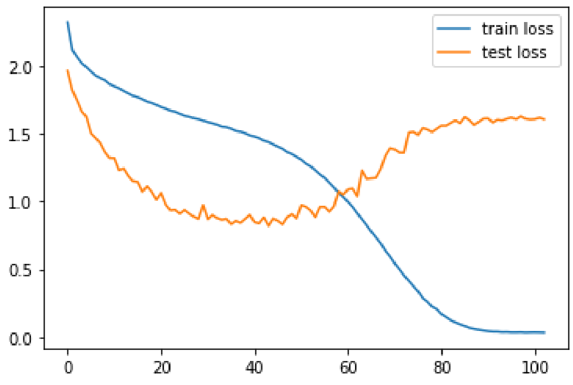


(a) Loss

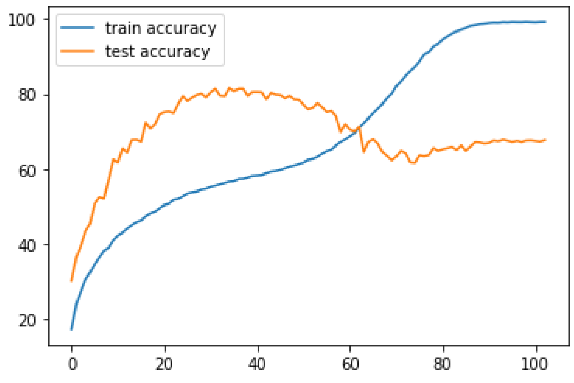


(b) Accuracy

Figure 56: Comparision of transfer learning on CIFAR100



(a) Loss



(b) Accuracy

Figure 57: Performance of student model trained with curriculum learning on CIFAR10 with 40% of label noise

and the data adding process stops at (0.4). We get a final test accuracy of 0.64, lower than the baselines reported in Zhou et al. (2021). However, the test accuracy shows a promising upward trend at the beginning but decreases when the noisy data is added. This is mainly because a dissatisfactory curriculum parameters. The k-NN probes are also not calibrated with temperature parameters.

4.4 Discussion

From a number of experiments, we demonstrate the effectiveness of paced learning when ambiguities between classes exist. The number of easy samples and the number of ambiguous samples should probably be large enough for CL to show its power. Currently, there is still a lack of naturally ambiguous examples for us to fully research this phenomenon. We also conclude the weaknesses of paced learning as follows. First, PL need precomputed difficulty scores which means training cannot be tuned end to end. Second, it takes expert knowledge to manually design the best combination of a scoring function and a pacing function as a useful curriculum. Third, although paced learning can reduce generalization gap, best performed curriculum parameters only come from exhaustive grid search at the moment. This makes curriculum learning a less efficient choice than sharpness aware minimization.

5 Automatic curricula

Despite its straightforward design, a manually designed curriculum can limit the performance of the training model due to issues discussed in Section IV. Unlike handcrafted curricula, learners have access to all materials in self-paced learning. However, it is worth to mention optimizing SPL can be unstable and tuning the hyperparameters can be a disaster in many circumstances. In this section, we opt to explore the reasons for these issues. An additive objective of this section is to tame an SPL algorithm for image classification that automatically creates curricula for our training model.

Let us split our dataset as training, validation and testing, and denote a training sample, its label and its empirical loss as x_i, l_i, y_i. SPL (Kumar et al., 2010) can be formalized as follows

$$\min_{w,v\in[0,1]^N} E(w,v;\lambda)[\sum_{i=1}^{N} v_i l_i + g(v;\lambda)] \tag{5.1}$$

$$v = \begin{bmatrix} v_1 & v_2 ... & v_N \end{bmatrix} \tag{5.2}$$

where v is a learnable weight vector of size N; λ, an aging function, models the learning stage or the age of our current model. The aging function is monotonically increasing with the number of iterations. The training objective 5.2 is mainly a empirical risk composed of weighted loss and a regularization over the weight vector. In the original SPL, the regularization term in (Kumar et al., 2010) is defined as

$$g(v;\lambda) = -\lambda(t)\sum_{i=1}^{N} v_i \tag{5.3}$$

To solve this multivariable optimization problem, SPL methods usually use an alternative optimizing strategy (AOS) that alternately fix one parameter while optimizing the other. When w is fixed, the objective becomes a linear term of v which is a convex function. Therefore the global optimum v^* can be analytically computed as

$$v_i{}^* = \begin{cases} 1, & l_i < \lambda \\ 0 & otherwise \end{cases} \tag{5.4}$$

Through these mathematical analysis, the following properties can be found. First, λ acts as a hard threshold function that dynamically adds in training samples over time. At the beginning of training, λ is small and only allows easier samples with small loss to attend training. As λ grows larger, more difficult samples are introduced and our model can learn more sophisticated knowledge from them. This age parameter λ is predefined, but unlike pacing functions which is problem agnostic it interacts intensively with loss. This implicitly shows the flexibilty of SPL because the thresholding can be applied on not only loss but other forms of feedback from from

the current model. Second, alternatively optimizing w and v will reduce the loss while maintaining (recovering) the distribution of training data. Unfortunately, the optima cannot be guaranteed due to instability of AOS Kumar et al. (2010).
The structure of this section is as follows. We investigate the behaviour of self-paced

Table 5: Commonly used regularisation terms

Regularizers	$g(\boldsymbol{v};\lambda)$	$v_i^*(l_i;\lambda)$
Hard	$-\lambda\sum_{i=1}^{N} v_i$	$\begin{cases}1, & l_i<\lambda\\ 0, & \text{otherwise}\end{cases}$
Linear	$\frac{1}{2}\lambda\sum_{i=1}^{N}(v_i^2-2v_i)$	$\begin{cases}1-l_i/\lambda, & l_i<\lambda\\ 0, & \text{otherwise}\end{cases}$
Logarithmic	$\sum_{i=1}^{N}\left(\zeta v_i-\frac{\zeta^{v_i}}{\log\zeta}\right)$ $\zeta=1-\lambda,\ 0<\lambda<1$	$\begin{cases}\dfrac{\log(l_i+\zeta)}{\log\zeta}, & l_i<\lambda\\ 0, & \text{otherwise}\end{cases}$
Mixture	$-\zeta\sum_{i=1}^{N}\log\left(v_i+\frac{\zeta}{\lambda_1}\right)$ $\zeta=\frac{\lambda_1\lambda_2}{\lambda_1-\lambda_2}$, $\lambda_1>\lambda_2>0$	$\begin{cases}1, & l_i\le\lambda_2\\ 0, & l_i\ge\lambda_1\\ \zeta\left(\dfrac{1}{l_i}-\dfrac{1}{\lambda_1}\right), & \text{otherwise}\end{cases}$

learning starting with a toy example, i.e., a logistic regression task on a 2D space. Next, we try out different loss-based curricula with training schedulers and assess their effects. Then we apply self-paced learning to image classification on CIFAR10H datasets and compare the results.

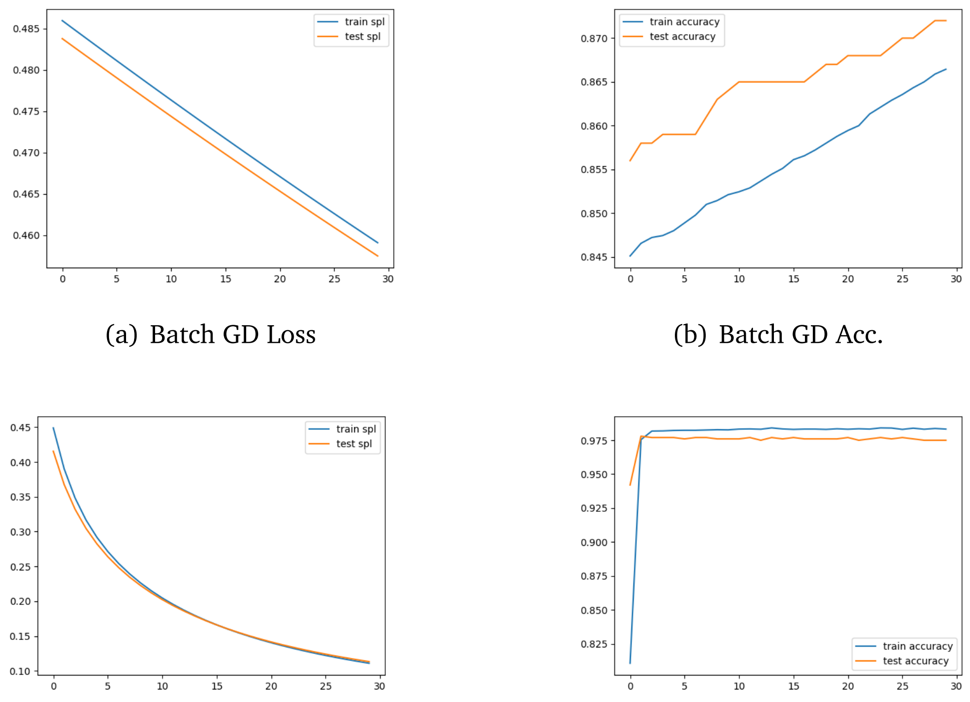
5.1 Class-imbalanced learning behavior

To illustrate the training behaviour of SPL, let us consider a binary logistic regression on a 2D space. The likelihood function can be formulated as follows

$$P(Y|W,X)=\prod_{i=1}^{N}P(\hat{y}_i=y_i|W,x) \tag{5.5}$$

$$\log P(Y|W,X)=\sum_{i=1}^{N}y_i\log(w_1x_{1i}+w_2x_{2i}+w_0)+(1-y_i)\log(1-(x_{1i}+w_2x_{2i}+w_0)) \tag{5.6}$$

For simplicity, we use a linear classifier because difficulty scores can be easily measured with distances to current decision boundary. The weights are initialized with Xavier initialization with a fixed random seed. Two clusters of data are generated on (0.,0.) and (3.,3.). We use the original loss function and the age parameter grows geometrically. To enforce fair comparison, we keep using the same geometric aging function for the following experiments. In Figure59, we observe the SPL strategy



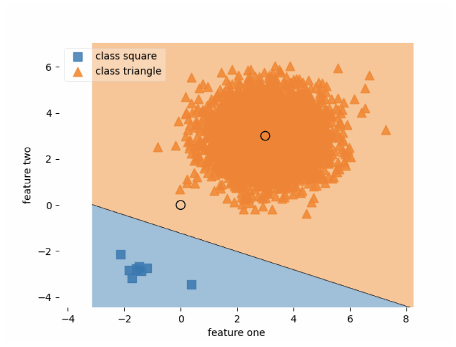
(a) Batch GD Loss (b) Batch GD Acc.

(c) SGD Loss (d) SGD Acc.

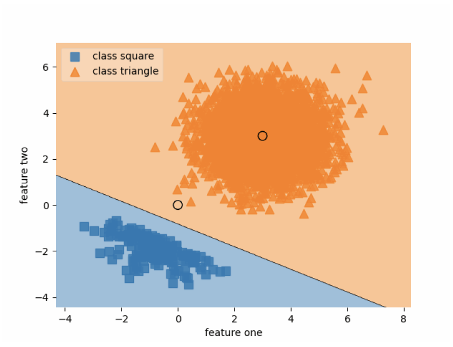
Figure 58: Comparison of SPL trained with Batch GD and SGD

first let the model perfectly classify examples of the orange class and then add in examples of the orange class. With a fixed learning rate as 0.005, we test training with stochastic gradient descent and batch gradient descent for 30 epochs. Stochastic gradient descent shows faster convergence on this binary classification problem (Figure58).

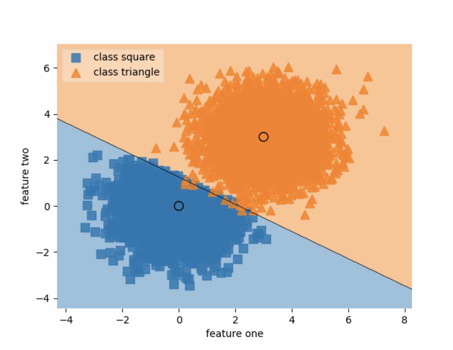
Then we test different regularization terms on the half-moon dataset with two-layer MLP. These soft regularisation terms extend the weight vector from binary values to continuous values between zero and one. Figure 63 shows examples that are far away from the decision boundary have higher losses coloured darker than those that are closer to the decision boundary. Figure 60, Figure 61 and Figure 62 show the dynamics of training with different regularization terms. Here we report weighted loss for SPL with soft regularisation terms.The weighted loss curves show overshoots at the early stage of training. As the model learns from more data, it increases the weights from samples. Therefore, the loss curves will first increase and then converge. We find the hard regularisation show comparable or even better performance.



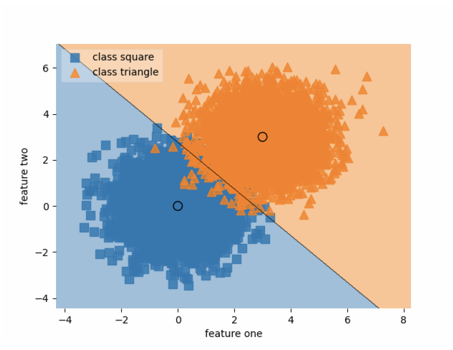
(a) 1st epoch



(b) 2nd epoch

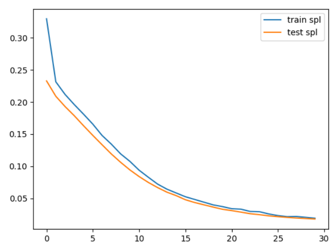


(c) 3rd epcoh

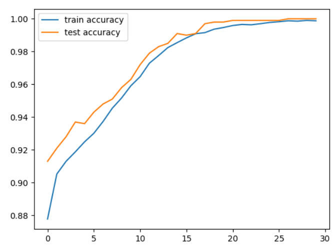


(d) 5 epoch

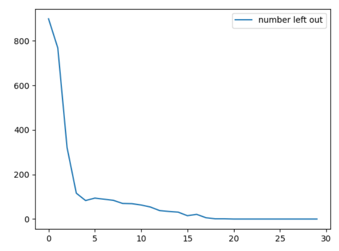
Figure 59: Self-paced learning with Hard regularization term on two Gaussians



(a) Loss

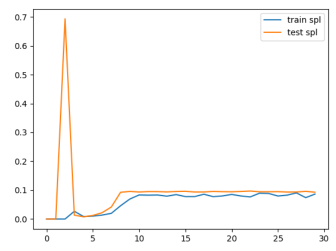


(b) Accuracy

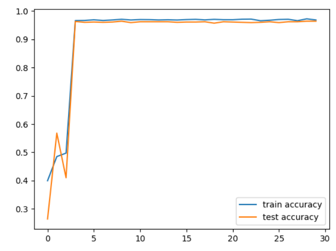


(c) Left out

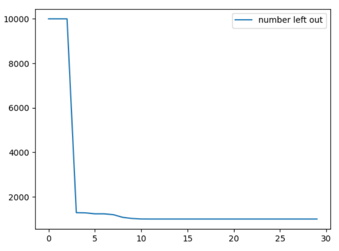
Figure 60: Training dynamics of SPL with hard regularisation



(a) Loss

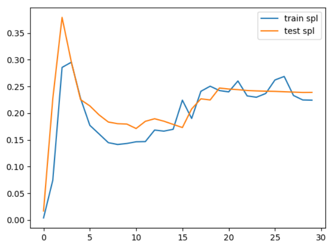


(b) Accuracy

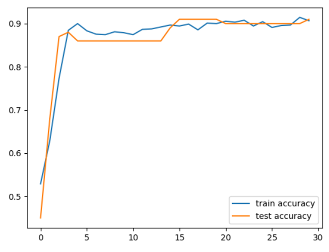


(c) Left out

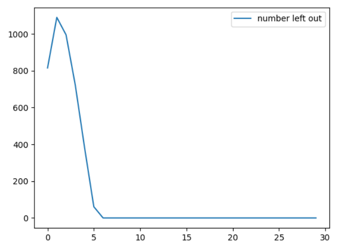
Figure 61: Training dynamics of SPL with linear regularisation



(a) Loss

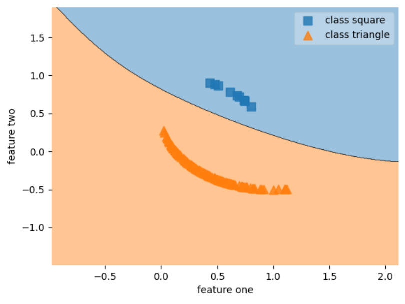


(b) Accuracy

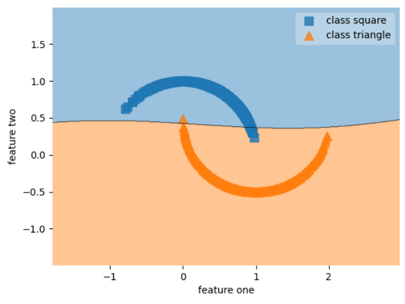


(c) Left out

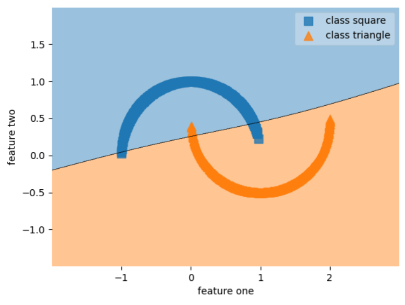
Figure 62: Training dynamics of SPL with log regularisation



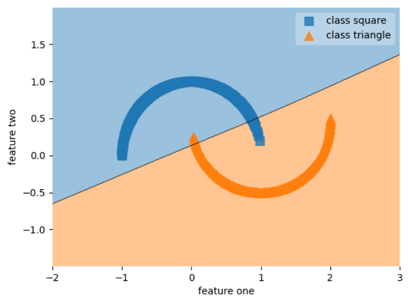
(a) 1st epoch



(b) 2nd epoch



(c) 3rd epcoh



(d) 5 epoch

Figure 63: Self-paced learning with aging regularization term (log) on Half moon dataset

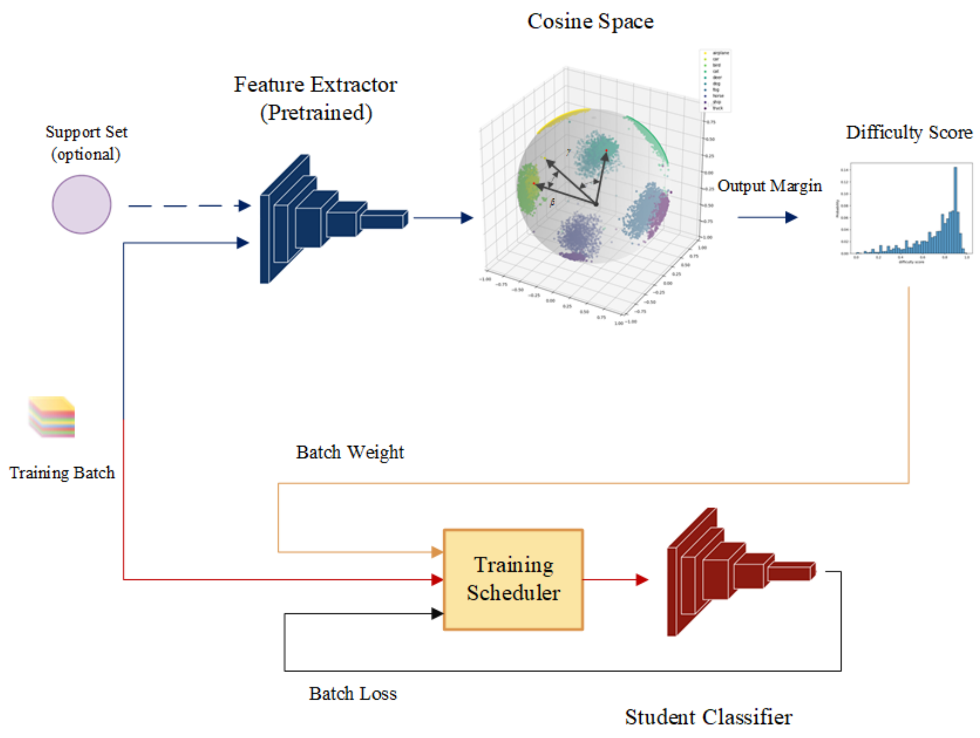
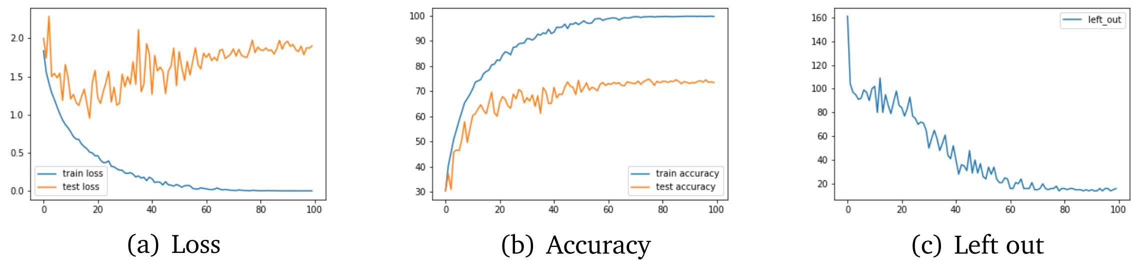


Figure 64: Self-paced curriculum learning with angular gap

5.2 SPL and SPCL on image classification

In this subsection we explore SPL and SPCL by testing whether students can progress more efficiently with teachers' initial guidance and constant supervision from labels. Figure 64 shows an example of how SPCL works with angular gap difficulty scores. The pretrained teacher network extracts features from the current batch and send the difficulty scores auxiliary signal to self-paced learning. This signal works together with the feedback from the student network and jointly create a dynamic curriculum.

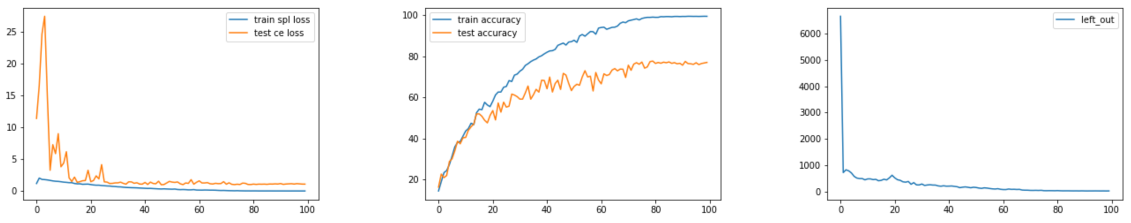
Figure 65 and Figure 66 show typical examples of SPL with geometric pacing func-



(a) Loss (b) Accuracy (c) Left out

Figure 65: SPL with geometric aging function

tions and linear functions. We choose a linear aging function with an initial threshold as b and increase λ linearly to the maximum loss threshold. We consider 20 as a large enough value for the loss of clean data. So the loss threshold starts with b and



(a) Loss (b) Accuracy (c) Left out

Figure 66: SPL with linear aging function

gradually increases until *a* percentage of total iterations when it reaches the maximum. For SPCL, we normalize the precomputed difficulty scores to values between 0 and 1, and plug them into aging parameter update rules. We create similar grid with Figure 67 to search the best curriculum learning for SPL and SPCL. We compare the best performed models trained with SPCL in Table 6 and Table 7. We observe SPCL achieves higher performance with C-score and PD. We guess the reason is likely to lie smoothness as both two scores have smoother distribution than others. This may lead to better solutions in alternative optimization.

Table 6: Performance on CIFAR10H

	Paced Learning		SPCL	
	Top1 Acc.	Top5 Acc.	Top1 Acc.	Top5 Acc.
Human Score	78.10	98.49	77.60	98.23
Forgetting events	77.43	98.14	76.80	97.92
C-Score	**78.15**	**98.67**	**78.10**	**98.50**
Prediction Depth	**78.26**	**98.79**	**78.15**	**98.66**
Angular Output Margin	77.93	98.50	77.51	97.75

Table 7: Performance on CIFAR100

	Paced Learning		SPCL	
	Top1 Err.	Top5 Err.	Top1 Err.	Top5 Err.
Forgetting events	22.91	7.68	22.91	7.12
C-Score	**22.02**	**5.96**	**22.50**	**6.33**
Prediction Depth	**21.86**	**5.90**	**22.68**	**6.45**
Angular Output Margin	22.50	6.33	22.85	6.50

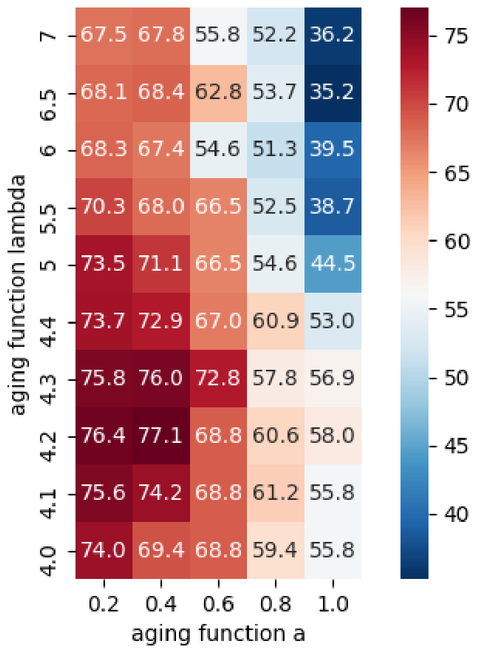


Figure 67: SPL search grid

6 Future work

Currently, we schedule to improve curriculum learning at two parts. First, we will increase model depth and find better calibration techniques to get more fine-grained prediction depth. With improved difficulty score, we can potentially increase generalizability. Second, pre-trained teacher networks provide guidance in the form of orders/ weights. The guidance is not strong enough. We opt to combine curriculum learning with knowledge distillation and see if two is better than one. There are a few works, (Xiang et al., 2020),(Aguilar et al., 2020), in this direction with similar ideas but their results are about 84 on CIFAR100 which is far from perfect. We are going to start this work from paced learning first.

7 Conclusion

We have come to the following conclusions through a large amount of curriculum learning experiments on image classification.

1. We highlight the importance of model calibration in difficulty measurement. For handcrafted curricula, a good difficulty scoring function should reflect example ambiguity accurately enough which means distinguishable for sorting. Furthermore, Prediction depth is a novel difficulty metric with great potential.

2. With intensive grid search, we demonstrate curriculum learning can increase deep model's generalisability for image classification. Presenting examples from easy to hard encourage the model to learn a simple function first. We further encourage practitioners to try our class-balance-aware paced learning scheduler. Optimizing a curriculum is limited to exhaustive grid search at the moment.

3. We propose a new difficulty metric called angular gap. Although this metric might be ”not among the best”, people can easily interpret example difficulty from the learnt hyper ball. can be easily This metric works reasonably well in both handcrafted and automatic curricula and can be further refined.

4. For self-paced learning, we find the original hard regularisation term performs reasonably well for clean datasets. In this project, automatic curricula did not outperform handcrafted ones for image classification on CIFAR100. In SPL, We observe the model first learns from an easy class and then moves to difficult ones.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In **Proceedings of the AAAI Conference on Artificial Intelligence**, volume 34, pages 7350–7357, 2020. pages 65

Mohammad Alsharid, R. El-Bouri, Harshita Sharma, L. Drukker, A. Papageorghiou, and J. Noble. A curriculum learning based approach to captioning ultrasound images. **Medical ultrasound, and preterm, perinatal and paediatric image analysis**, 12437:75–84, 2020. pages 18

Robert JN Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. **arXiv preprint arXiv:2106.09647**, 2021. pages 13, 16, 27

R. Battleday, Joshua C. Peterson, and T. Griffiths. Capturing human categorization of natural images by combining deep networks and cognitive models. **Nature Communications**, 11, 2020. pages 7, 8

Yoshua Bengio, J. Louradour, Ronan Collobert, and J. Weston. Curriculum learning. In **ICML '09**, 2009. pages 6, 19, 41

Jerome S Bruner, Jacqueline J Goodnow, and George A Austin. **A study of thinking**. Routledge, 2017. pages 5

Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. Visualizing and understanding curriculum learning for long short-term memory networks, 2016. pages 22

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition, 2019. pages 35

Yixiao Ge, Da peng Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Self-paced contrastive learning with hybrid memory for domain adaptive object re-id. **ArXiv**, abs/2006.02713, 2020. pages 22

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. **ArXiv**, abs/1706.04599, 2017. pages 23

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. pages 36, 37

Mobarakol Islam and B. Glocker. Spatially varying label smoothing: Capturing uncertainty from expert annotations. In **IPMI**, 2021. pages 24

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, S. Shan, and A. Hauptmann. Self-paced learning with diversity. In **NIPS**, 2014. pages 22

Lu Jiang, Deyu Meng, Qian Zhao, S. Shan, and A. Hauptmann. Self-paced curriculum learning. In **AAAI**, 2015. pages 22

Lu Jiang, Zhengyuan Zhou, Thomas Leung, L. Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In **ICML**, 2018. pages 22

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions, 2019. pages 18

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and M. Mozer. Characterizing structural regularities of labeled data in overparameterized models. **arXiv: Learning**, 2020. pages 14, 39

Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Self-paced deep reinforcement learning, 2020. pages 22

M. Kumar, Ben Packer, and D. Koller. Self-paced learning for latent variable models. In **NIPS**, 2010. pages 22, 58, 59

Manish Kumar, George F. Foster, Colin Cherry, and M. Krikun. Reinforcement learning based curriculum optimization for neural machine translation. In **NAACL**, 2019. pages 6, 22

Yutian Li, Feng Gao, Zhijian Ou, and Jiasong Sun. Angular softmax loss for end-to-end speaker verification. In **2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)**, pages 190–194. IEEE, 2018. pages 35

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In **Proceedings of the IEEE conference on computer vision and pattern recognition**, pages 212–220, 2017. pages 35

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In **NeurIPS**, 2019. pages 24, 37

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020. pages 6, 22

Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In **CVPR Workshops**, volume 2, 2019. pages 23

Giorgio Patrini, A. Rozza, A. Menon, R. Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 2233–2241, 2017. pages 37

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. Curriculum learning of multiple tasks, 2014. pages 22

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. **arXiv preprint arXiv:1706.05806**, 2017. pages 16

Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination, 2016. pages 18

Daniel Soudry, E. Hoffer, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. **ArXiv**, abs/1710.10345, 2018. pages 13

Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In **Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics**, pages 751–759, 2010. pages 20

Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. On the geometry of generalization and memorization in deep neural networks. **arXiv preprint arXiv:2105.14602**, 2021. pages 16

Mariya Toneva, Alessandro Sordoni, Rémi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. **ArXiv**, abs/1812.05159, 2019. pages 13, 47

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition, 2018. pages 35

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning, 2021. pages 20

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In **European conference on computer vision**, pages 499–515. Springer, 2016. pages 18

X. Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? **ArXiv**, abs/2012.03107, 2020. URL `https://github.com/google-research/understanding-curricula/blob/9e6774b48587b4a24affc61edec3d524ac378aa7/main_w_test.py`. pages 6, 22, 41, 47, 50

Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In **European Conference on Computer Vision**, pages 247–263. Springer, 2020. pages 65

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. An empirical exploration of curriculum learning for neural machine translation, 2018. pages 22

Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Robust curriculum learning: from clean label detection to noisy label self-correction. In **International Conference on Learning Representations**, 2021. URL `https://openreview.net/forum?id=lmTWnm3coJJ`. pages 6, 57