Unsupervised dataset distillation

Project

- General task: train a (shallow) feature extractor via PSRs (on a learned dataset) that work well with a task-specific head

$$\min_{\mathcal{D}} \sum\nolimits_j \min\nolimits_{w_{\text{head}}^j} \mathcal{L}_{\text{tr}}^j(w_{\text{head}}^j \circ w_{\text{base}}^*) \qquad \text{s.t. } w_{\text{base}}^* = \arg\min_w \mathcal{L}_{\text{PSR}}(w; \mathcal{D})$$

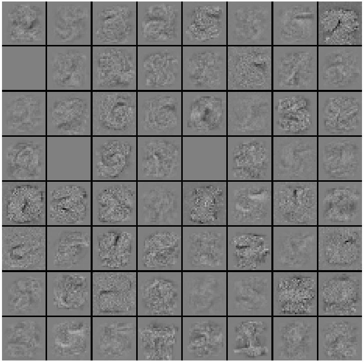
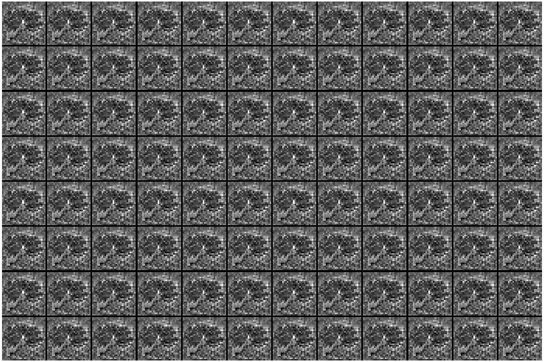
- Why shallow feature extractor?
  Inner problem should be “easy” and quick to solve
  Head part of NN can be arbitrary
- E.g. use shallow LVM such as GRBM

$$\log p_\theta(x) \doteq -\tfrac{1}{2\sigma^2}\|x-a\|^2 + \sum\nolimits_k S(w_k^\top x + b_k)$$
$$\nabla_x \log p_\theta(x) = \tfrac{1}{\sigma^2}\|a-x\|^2 + \sum\nolimits_k w_k \sigma(w_k^\top x + b_k)$$

  Feature extractor $S'(w_k^\top x + b_k) = \sigma(w_k^\top x + b_k)$
- Dataset $\mathcal{D} = \{x_1, \dots, x_N\}$ of unlabeled data

Quadratic penalty method

Epoch	Test accuracy	$\|X - AE(x)\|$
1	46.1%	5.4
100	92.1%	0.067
200	94.1%	0.01

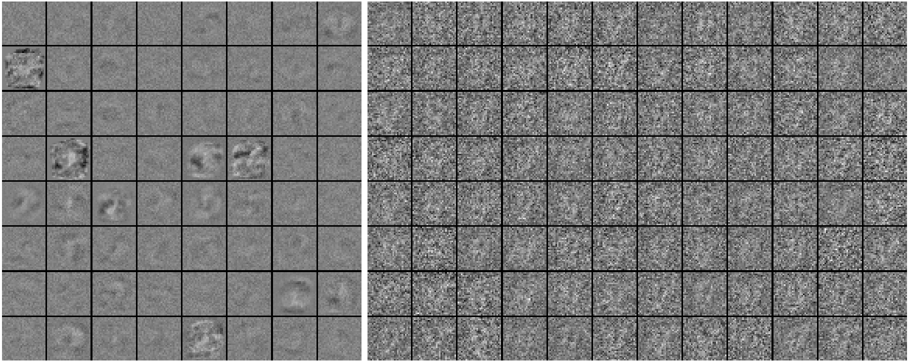



w_{base} and distilled dataset $\mathcal{D}$

Unsupervised dataset distillation: Reference implementation

Quadratic penalty method (now with "diversity term" $-\lambda_{\text{divers}}\|\mathcal{D} - \mathrm{mean}(\mathcal{D})\|^2$)

Epoch	Test accuracy	$\|X - AE(x)\|$
1	24.0%	5.6
100	92.3%	0.22
200	93.2%	0.20

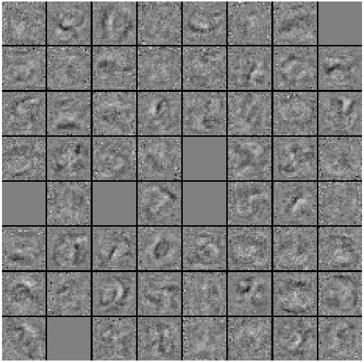


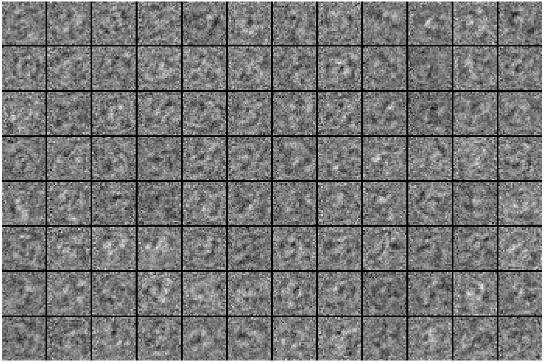
w_{base} and distilled dataset $\mathcal{D}$

Unsupervised dataset distillation: Reference implementation

Optimal value reformulation (diversity term has little impact)

Epoch	Test accuracy	$\|X - AE(x)\|$
1	32.6%	4.8
100	81.4%	2.6
200	94.6%	3.3





w_{base} and distilled dataset $\mathcal{D}$

Unsupervised dataset distillation: What you should do

- Choose meta-learning approach
  - Quadratic penalty method (probably the easiest to make it work)
  - Optimal value reformulation
  - Implicit differentiation & approximate Hessian matrix inversion
  - Or any other method that approximately solves bilevel programs
- Baseline code provided
  - Available on Canvas
  - Implements penalty methods using AE as unsupervised learner
  - Implemented in Julia + Flux
- Extend what is implemented in the baseline, e.g.
  - Different NN / DEM architectures, different target dataset and/or different $|\mathcal{D}|$
  - Multiple training datasets for outer loss (MTL setting)
  - Use different PSRs / unsupervised learning methods (instead of simple AE)
  - Investigate other ways to prevent mode collapse and/or to make $\mathcal{D}$ look more natural
    - E.g. pre-train a prior $\log p_\psi(x)$ to represent a desired image statistics and add to loss
    - Note: there is an ambiguity between w_{base} and $\mathcal{D}$: $w_{\text{base}}\mathcal{D} = w_{\text{base}}QQ^{-1}\mathcal{D}$
  - Apply the approach layer-wise / meta-learn multiple layers at once
  - You only need to work on some of these (and your own) suggestions, not all!
    - Your time spent should be worth 2 ECTS ($\approx$ 40 hours, i.e. 56 hours minus lecture time)

Unsupervised dataset distillation: Hints

- If you opt for the quadratic penalty method, you need to calculate $\nabla_w \mathcal{L}_{\text{inner}}(w, \theta)$
  - Because most auto-diff packages cannot do $\nabla_w f(\nabla_x g(x, w))$ well (but JAX might be able to do it)
- In the case of AE and GBRM:

$$\nabla_{W,b,a} \left(\frac{1}{2} \left\| \frac{1}{\sigma^2}(a - x) + W\sigma(W^\top x + b) \right\|^2 \right)$$

- Define $\mathbf{r} := \frac{1}{\sigma^2}(a - x) + W\sigma(W^\top x + b)$
- Main task: ∇_{w_k}, where w_k is k-th column of W

$$\nabla_{w_k} \left(\frac{1}{2} \left\| \frac{1}{\sigma^2}(a - x) + \sum w_k \sigma(w_k^\top x + b_k) \right\|^2 \right)$$

- Hint: ∇ yields column vector, hence chain rule is transposed

$$\nabla_{w_k} = \left(\sigma(w_k^\top x + b_k)\mathrm{I} + \sigma'(w_k^\top x + b_k) w_k x^\top \right) \mathbf{r}$$

- Efficient implementation in Julia:

```
as = m.W' * Xs .+ m.b
hiddens = my_σ.(as); dhiddens = ∇my_σ.(as)
XXs = m.W * hiddens .+ m.a / σ2; residual = (XXs - Xs / σ2) / N
W_r = m.W' * residual

gs_W = residual * hiddens' + Xs * (W_r .* dhiddens)'
gs_b = sum(W_r .* dhiddens, dims=2)
gs_a = sum(residual, dims=2) / σ2

return sum(abs2.(gs_W)) + sum(abs2.(gs_b)) + sum(abs2.(gs_a))
```

Unsupervised dataset distillation: Hints

- Consult the "Matrix Cookbook" for vector and matrix-based calculus
- Penalty: verify numerically that $\|\nabla_w \mathcal{L}_{\text{inner}}(w, \theta)\|^2$ matches with and w/o auto-diff
  - Math: Jacobian $\sigma'(w_k^\top x + b_k)$ is a diagonal matrix (tensor for a batch)
  - Implementation: `dhiddens = ∇my_σ.(W'*x + b)` is a vector (matrix for a batch)
  - Efficient summation over entire batch
- OVR: make sure $\min_{w'} \mathcal{L}_{\text{inner}}(w', \theta)$ works well!

Don't panic

- Solving bilevel programs is difficult & brittle!
  - Especially for non-convex inner losses
  - Results may vary significantly with different hyper-parameters
- No need to win a benchmark!

Unsupervised dataset distillation: What to submit

- Short report (pdf!) describing what you have done beyond the baseline code
  - And some visualization of the distilled dataset and the training progress
  - Also document unexpected problems/surprises you encountered
- Your Python / Julia / Matlab / . . . code
- Email to `zach@chalmers.se` by November 30, 2024
  - With subject containing "Project LFR-2024"
- Contact me with a proposal for a project if you cannot do this one
  - E.g. no access to sufficiently capable PC or no coding experience

You can work in pairs (and submit one pdf and implementation)!

Examples of last year's project reports

WASP Learning Feature Representations
Module 1 Assignment

1. CNN acrchitecture

We consider two network architectures. "Conv4" is a simple convolutional network with four blocks, each consisting of a convolutional layer followed by batch normalization, a ReLU activation function and average pooling. "ResNet" is a residual network with 9 ResNet blocks. Each ResNet block consists of two layers of convolution / batch normalization / ReLU. Each network has a simple fully-connected head.

2. Backbone pre-training

We pre-train the backbones on the Tiny Imagenet dataset using SimiCLR [1], a contrastive method for representation learning. This method learns representations by maximizing the agreement between differently augmented views of the same image in the latent space using the InfoNCE loss [2].

Each image is randomly augmented twice to produce two different views of the same example. Following the SimCLR paper, we use *random cropping*, *random color distortion* and *random Gaussian blur*. We also use *random horizontal flip* and *random greyscale*. Fig 1 shows some examples of augmented pairs.

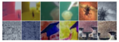


Figure 1. **Data augmentation**. Augmented image examples from the Tiny Imagenet dataset.

As illustrated in Fig. 2, the two augmented views are encoded by an *encoder network* f (the backbone of the network that we want to pre-train) to generate the representations $\mathbf{h}_i$ and $\mathbf{h}_j$. These representations are subsequently transformed by a *projection head* g, yielding $\mathbf{z}_i$ and $\mathbf{z}_j$. This projection head is a two-layer MLP, which is used in place of the network's head during pre-training.

The network is trained to maximize the agreement between $\mathbf{z}_i$ and $\mathbf{z}_j$ using the loss

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \tag{1}$$

where $\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k) = \frac{\boldsymbol{z}_i^T \cdot \boldsymbol{z}_k}{\|\boldsymbol{z}_i\| \cdot \|\boldsymbol{z}_k\|}$ is the cosine similarity between vectors $\boldsymbol{z}_i$ and $\boldsymbol{z}_k$, and τ is a temperature parameter. This loss encourages the network to pull closer the representations of positive pairs $(\mathbf{z}_i, \mathbf{z}_j)$, while pushing away negative pairs (i.e. all other feature vectors $\boldsymbol{z}_k$ in the mini-batch).

1

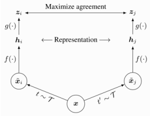


Figure 2. **The SimCLR framework**. A backbone network f and projection head g are trained to maximize the agreement between two augmented views of the same image using a contrastive loss (image from [1]).

3. Evaluation on downstream tasks

After pre-training, the projection head is discarded and the backbones are used for evaluation on downstream tasks on the CIFAR10 dataset.

3.1. Fully-supervised baseline

In the following experiments, all fully-supervised baselines are trained from scratch for 180 epochs using SGD with momentum 0.9, weight decay 0.0001 and batch size 128. The initial learning rate is 0.1, and is multiplied by 0.1 at epochs 100 and 150.

3.2. Linear evaluation

We first evaluate the quality of the representations learned during pre-training under the linear classification protocol. We freeze the backbone weights and train only the network's final linear layer. We train for 180 epochs using SGD with momentum 0.9, weight decay 0.001 and batch size 64. The initial learning rate is 0.001, and is multiplied by 0.1 at epochs 100 and 150. We report the mean and standard deviation of the Top-1 accuracies obtained when training the models with four random seeds.

Backbone	Pre-trained	Supervised
Conv4	48.74 ± 0.09	79.44 ± 0.28
ResNet	52.91 ± 0.19	90.89 ± 0.12

Table 1. **Linear evaluation protocol.** Performance comparison (Top-1 accuracy, %) with the fully-supervised baseline on CIFAR10.

As shown in Table 1, the linear layers trained on the frozen backbones achieve an accuracy that is far better than random (which would be 10% on CIFAR10). However, we still observe a significant gap with the fully-supervised baseline.

3.3. Semi-supervised learning

We now evaluate the learned representation on a semi-supervised learning task. We sample 1%, 5% and 10% of the CIFAR10 training set as labeled samples and fine-tune the networks (both the backbones and the linear heads). We train for 180 epochs using SGD with momentum 0.9, weight decay 0.0001 and batch size 128. The initial learning rate is 0.1, and is multiplied by 0.1 at epochs 100 and 150. We report top-1 accuracy on the entire test set. We present results averaged over four random seeds.

	Conv4		ResNet	
Label fraction	Fine-tuned	Supervised	Fine-tuned	Supervised
1%	45.54 ± 0.8	40.15 ± 0.5	52.62 ± 1.1	38.04 ± 0.8
5%	60.95 ± 0.2	57.98 ± 1.0	69.73 ± 0.4	62.94 ± 0.8
10%	66.23 ± 0.5	65.22 ± 0.4	75.29 ± 0.4	73.26 ± 0.6
50%	76.25 ± 0.3	76.37 ± 0.4	87.23 ± 0.1	86.68 ± 0.4
100%	78.99 ± 0.1	79.44 ± 0.3	90.93 ± 0.1	90.89 ± 0.1

Table 2. **Semi-supervised learning on CIFAR10** with different label fractions for fine-tuning. We present results averaged over four random seeds.

As shown in Table 2, fine-tuning the pre-trained networks performs better than training from scratch when using fewer labeled samples. However, when fine-tuning on the full training set the pre-trained networks do not perform better than the networks trained from scratch.

Note however that we did not perform (extensive) hyperparameter search for fine-tuning; the hyperparameters are roughly the same as when training from scratch.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. Feb. 2020.
[2] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. July 2018.

Examples of last year's project reports

Learning Feature Representations Assignment Module 1

Abstract

In the context of this assignment, we explored different unsupervised pre-training techniques and investigated their benefit on the image-classification downstream task. More specifically, we considered the Deep InfoMax framework as well as plain denoising-AE for unsupervised pre-training purposes under different setups of backbone tuning and different amounts of available data during the fully-supervised training. Our results suggest that the unsupervised pre-training can boost classification performance in low data regimes and in the absence of backbone fine-tuning. Conversely, the benefits of our considered pre-training setups are not as evident in the backbone fine-tuning setup. The code can be found at https://github.com/

1 Problem Definition

This project aims to leverage unsupervised pre-training approaches and evaluate whether and to which extent they improve the performance in the image classification task using deep neural network (DNN) architectures. Given a labeled image dataset A and an unlabeled one B, at first, we perform unsupervised pre-training on the latter. We then use the learned pre-trained weights to initialize the DNN network and fine-tune on dataset A.

2 Methodology

2.1 Baseline

In order to measure the merit of using pre-training, a suitable baseline architecture is needed. For that, we use the encoder architecture as found in DuaneNielsen. More specifically, the architecture consists of an encoder of four convolutional layers and a fully connected one paired with three fully connected layers classification head. The architecture's weights are updated towards minimizing the cross entropy loss between the prediction and the target.

2.2 Denoising Autoencoder

A denoising Autoencoder (dAE) architecture was trained to reconstruct samples perturbed with Gaussian noise $\epsilon \sim N(0,0.1)$. The dAE utilized the baseline's encoder architecture whereas the decoder is a mirrored version of that same encoder. Here, the weights are updated towards minimizing the mean squared error between the reconstructed and the unperturbed input. Assuming an unperturbed input X and a encoder-decoder architecture, the reconstruction loss can be formulated as:

$$L_{\text{RE}} = X - \overline{X_p} \tag{1}$$

$$\text{with } \overline{X_p} = \text{Dec} \circ \text{Enc}(X + \epsilon), \tag{2}$$

2.3 Deep InfoMax

Mutual information (MI) is a measure of independence indicating the amount of information revealed for one random variable X by knowing the other Y. From a representation learning perspective, we are interested in generating informative latent representation of the data in order to address downstream tasks such as image classification and speech recognition etc. Assuming X and Y to be the data and their corresponding latent representations respectively, a high mutual information between these two random variables is desired, since Y conditioned on X - $P(Y \mid X)$ - would reveal a lot of information about the latter. On the contrary, a non-extend (0) MI between X and Y would mean that $P(Y \mid X) = P(Y)$, or in other words that Y does not enclose any information about X. MI can be expressed as the Kullback–Leibler divergence between the join distribution $P(X,Y)$ and the product of the marginals $P(X)P(Y)$:

$$I(X,Y) = D_{KL}[P(X,Y) \,||\, P(X)P(Y)] \tag{3}$$

Having explained the motivation on why having high MI between the input and their corresponding latent representation is a desirable property, we adopt the Deep InfoMax (DIM) Hjelm et al. (2018) framework to carry out for unsupervised pre-training purposes. Conceptually, DIM aims at learning useful representation based on two main ideas, (i) Maximizing the MI between the low-level and its high-level feature representation of the input, and (ii) constraining the generated representation to follow an assumed prior distribution.

2.3.1 Global Deep InfoMax

To realize the DIM objectives, the baseline Encoder architecture E_ψ, parametrized by ψ is decomposed between the low-level C_ψ of $M \times M$ dimension and high-level f_ψ feature extraction modules as $E_\psi = f_\psi \circ C_\psi$. In DIM, the MINE Belghazi et al. (2018) framework is utilized in order to estimate the MI. More specifically, MINE uses the Donsker Varadhan (DV) Donsker & Varadhan (1983) KL divergence formulation in order to compute the lower bound of the MI as:

$$\mathcal{I}(X;Y) := \mathcal{D}_{KL}[P(X,Y) \,||\, P(X)P(Y)] \geq \widehat{\mathcal{I}}^{(DV)}_\omega(X;Y) \tag{4}$$

$$= \mathbb{E}_{P(X,Y)}[T_\omega(x,y)] - \log \mathbb{E}_{P(X)P(Y)}[e^{T_\omega(x,y)}], \tag{5}$$

where $T_\omega : X \times Y \to \mathbb{R}$ is a discriminator network parameterized by ω.

Considering that both the discriminator T and the encoder architecture aim at maximizing the MI objective we can alleviate the computation by sharing layers between these modules. Based on that, the discriminator architecture is realized as $T_{\psi,\omega} = D_\omega \circ g \circ f_\psi \circ C_\psi$ with D being the classification head and the g being a function combining the high and low feature representation. Building upon the observation that the exact value of the MI is not of interest but rather the minimization of that, the DIM adopts different divergences with more favorable properties. In this work, we considered the Jensen–Shannon (JSD), and therefore the global MI Iglobal can be expressed as:

$$\widehat{\mathcal{I}\text{global}}^{(\text{JSD})}_{\omega,\psi}(C_\psi(X); E_\psi(X)) := \mathbb{E}_{x\sim\mathbb{P}}[-\text{sp}(-T_{\psi,\omega}(C_\psi(x), E_\psi(x)))] \tag{6}$$

$$-\mathbb{E}_{(x,x')\sim\mathbb{P}\times\tilde{\mathbb{P}}}[\text{sp}(T_{\psi,\omega}(C_\psi(x'), E_\psi(x)))], \tag{7}$$

where $sp(s)$ is the sopftplus function.

When computing the $\mathbb{E}_{x\sim\mathbb{P}}[.]$ term of the objective, we sample (true) pairs of low-level and high-level feature representations that have been derived from the same input (joint) whereas when computing the $\mathbb{E}_{(x,x')\sim\mathbb{P}\times\tilde{\mathbb{P}}}[.]$ term we sample (false) pairs of low-level and high-level representations that have been generated from different samples (product of the marginals).

2.3.2 Local Deep InfoMax

The MI objective as formulated in the previous section maximizes the information between the extracted high-level representation and the whole low-level feature map and therefore is termed as

global MI objective. It is argued in Hjelm et al. (2018) that this formulation does not always result in desirable properties since the high-level representation will be encouraged to enclose irrelevant (e.g. background) information of the input. To mitigate this shortcoming, a local MI objective variation is constructed such that the average MI is maximized between the patches of the low-level feature map and the high-level representation. In this sense, the local MI objective encourages the high-level representation to enclose the information that is shared among all the local patches. The local MI is computed as:

$$\widehat{\mathcal{I}\text{local}}^{(\text{JSD})}_{\omega,\psi}(C_\psi(X); E_\psi(X)) := \frac{1}{M^2}\sum_{i=1}^{M^2} \widehat{\mathcal{I}\text{global}}^{(\text{JSD})}_{\omega,\psi}(C^{(i)}_\psi(X); E_\psi(X)). \tag{8}$$

2.3.3 Prior matching

DIM also enforces the push-forward distribution $U_{\psi,\mathbb{P}}$ of the encoder E_ψ to match a uniform distribution V. The prior matching is realized through an adversarial training approach. The distribution matching is realized through training a discriminator D_ϕ to estimate the divergence $\widehat{D_\phi}[V||U_{\psi,\mathbb{P}}]$ while the encoder E_ψ is trained to minimize that estimate. The prior matching objective can be expressed as:

$$\widehat{D_\phi}[V||U_{\psi,\mathbb{P}}] = \mathbb{E}_{y\sim\mathbb{V}}[\log D_\phi(y)] + \mathbb{E}_{y\sim U_{\psi,\mathbb{P}}}[\log(1 - D_\phi(y))] \tag{9}$$

Finally, we combine both global and local MI objectives and the prior matching into a single minimization objective we estimate the network based on:

$$\arg\max_{\omega_1,\omega_2,\psi}(\alpha\widehat{\mathcal{I}\text{global}}^{(\text{JSD})}_{\omega_1,\psi}(C_\psi(X); E_\psi(X)) + \beta\widehat{\mathcal{I}\text{local}}^{(\text{JSD})}_{\omega_2,\psi}(C_\psi(X); E_\psi(X)) + \arg\min_\psi \arg\max_\phi \gamma\widehat{\mathcal{D}}_\phi(\mathbb{V}||\mathbb{U}_{\psi,\mathbb{P}}), \tag{10}$$

Note that the global and local MI objectives use different discriminators $T_{\omega1}$ and $T_{\omega2}$ parametrized by ω_1 and ω_2 respectively. The hyperparameters α, β, and γ are tunable and weight the contribution of each term in the optimization process.

3 Experiments

In the context of this assignment, the CIFAR10 Krizhevsky et al. (2009) and the 32×32-resolution variant of TinyImageNet Le & Yang (2015) were utilized for supervised training and unsupervised pre-training respectively.

3.1 Implementation Details

We trained dAE and DIM for 10 and 20 epochs respectively whereas in both unsupervised training approaches a 64-dimensional high-level latent representation was used, a batch size of 128, and a learning rate of 1e-4. The baseline architecture was trained for 20 epochs with a learning rate of 1e-3 and a batch size of 64. Finally, the Adam optimizer was used for all three training setups. For DIM we set the hyperparameters α, β, γ to 0.5, 1, and 0.1 respectively.

3.2 Results

We experimented with two different unsupervised pre-training settings, namely dAE and DIM. We also considered the setups of (i) fine-tuning: training the encoder end-to-end or only training the encoder's head, (ii) fraction: either training on the 100% or 5% of the CIFAR10 images. In table 1 we report the classification performance on the CIFAR10 test split. In the no fine-tuning setting, the benefit of using unsupervised pre-training is evident since both dAE and DIM outperform the baseline architecture. In all setups, the DIM pre-training led to better classification performance apart

from the fine-tuning on the 100% of the data where the baseline performs slightly better. In all cases using 100% of the data resulted in better performance compared to only using 5%. Additionally, fine-tuning also resulted in better-performing models compared to not using fine-tuning.

Method	fraction	no fine-tune	fine-tune
Baseline	5%	22.46	39.13
	100%	31.22	**78.23**
Baseline+dAE	5%	31.98	39.26
	100%	44.06	78.21
Baseline+DIM (JSD)	5%	**38.29**	**43.00**
	100%	**48.29**	77.60

Table 1: Accuracy performance in CIFAR10 test set

3.3 Conclusions

In this assignment, we evaluated the effect of using unsupervised pre-training on the downstream task of image classification. In the low data regime both unsupervised pre-training approaches lead to better-performing models, with the DIM outperforming the dAE. When 100% of the data was used, the dAE perform similarly to the baseline architecture whereas the DIM outperformed the baseline in the no-fine-tuning setting but performs slightly worse when the baseline is trained end-to-end. Towards further investigating the aforementioned unintuitive behavior one could potentially re-run the experiments using different seeds, introduce a warm-up training period, train DIM for longer, or experiment with different hyperparameters setups. More specifically, increasing the contribution of the local MI objective seems to be more suitable for classification tasks according to the authors in Hjelm et al. (2018).

References

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.

Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983.

DuaneNielsen. Duanenielsen/deepinfomaxpytorch: Learning deep representations by mutual information estimation and maximization. URL https://github.com/DuaneNielsen/DeepInfomaxPytorch.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.