Lautum Regularization for Semi-Supervised Transfer Learning – Supplementary Material*

Daniel Jakubovitz, Raja Giryes
School of Electrical Engineering
Tel Aviv University
Tel Aviv, Israel
danielshaij@mail.tau.ac.il
raja@tauex.tau.ac.il

Miguel R. D. Rodrigues
Department of Electronic and Electrical Engineering
University College London
London, United Kingdom
m.rodrigues@ucl.ac.uk

A. Information theory definitions

Let X, Y be two random variables with respective probability density functions $p(x), p(y)$. We present the definitions of three information theoretic measures which are relevant to our derivations.

Definition 1 (Mutual information) *The mutual information between X and Y is defined by*

$$I(X;Y) = \iint p(x,y) \log \left\{ \frac{p(x,y)}{p(x)p(y)} \right\} dxdy. \quad (1)$$

The Mutual information captures the dependence between two random variables. It is the Kullback-Leibler divergence between the joint distribution and the product of the marginal distributions. The following is the Lautum information:

Definition 2 (Lautum information) *The Lautum information between X and Y is*

$$L(X;Y) = \iint p(x)p(y) \log \left\{ \frac{p(x)p(y)}{p(x,y)} \right\} dxdy. \quad (2)$$

This measure is the Kullback-Leibler divergence between the product of the marginal distributions and the joint distribution. Similar to the mutual information, the Lautum information is related to the dependence between two random variables. However, it has different properties than the mutual information, as outlined in [4]. The last definition is of the differential entropy of a random variable.

Definition 3 (Differential entropy) *The differential entropy of a random variable X is defined by*

$$H(X) = -\int p(x) \log p(x) dx. \quad (3)$$

*The first workshop on Statistical Deep Learning for Computer Vision, in *Seoul, Korea*, 2019.

B. Proof of Theorem 1

Let us reiterate Theorem 1 before formally proving it.
Theorem 1 *For a classification task with ground-truth distribution $p(y|x)$, training set $\mathcal{D}$, learned weights $w_{\mathcal{D}}$ and learned classification function $f(y|x, w_{\mathcal{D}})$, the expected cross-entropy loss of a machine learning algorithm on the test distribution is equal to*

$$\mathbb{E}_{w_{\mathcal{D}}} \left\{ KL(p(x,y)||f(x,y|w_{\mathcal{D}})) \right\} + H(y|x) - L(w_{\mathcal{D}}; x). \quad (4)$$

Proof. The expected cross-entropy loss of the learned classification function $f(y|x, w_{\mathcal{D}})$ on the test distribution $p(x, y)$ is given by

$$\mathbb{E}_{(x,y)\sim p(x,y)} \mathbb{E}_{w \sim p(w_{\mathcal{D}})} \{ -\log f(y|x, w_{\mathcal{D}}) \}. \quad (5)$$

Explicitly, (5) can be written as [1]

$$-\iiint p(x,y)p(w_{\mathcal{D}}) \log f(y|x, w_{\mathcal{D}}) \, dx \, dy \, dw_{\mathcal{D}}. \quad (6)$$

To compare the learned classifier with the true classification of the data we develop (6) further as follows:

$$= -\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{f(y|x, w_{\mathcal{D}})}{p(y|x, w_{\mathcal{D}})} p(y|x, w_{\mathcal{D}}) \right\} dx \, dy \, dw_{\mathcal{D}}. \quad (7)$$

Using standard logarithm arithmetic we get the follow-

[1] Since the values of y are discrete it is more accurate to sum instead of integrate over them. Yet, for the simplicity of the proof we present the derivations using integration.

ing expression:

$$= \underbrace{-\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{f(y|x,w_{\mathcal{D}})}{p(y|x,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}}}_{(\star)} \\ \underbrace{-\iiint p(x,y)p(w_{\mathcal{D}}) \log p(y|x,w_{\mathcal{D}})\, dx\, dy\, dw_{\mathcal{D}}}_{(\star\star)}. \tag{8}$$

We separate the derivations of the two terms in (8). First, we develop the term $(\star\star)$ further:

$$\begin{aligned}(\star\star) &= -\iiint p(x,y)p(w_{\mathcal{D}}) \log p(y|x,w_{\mathcal{D}})\, dx\, dy\, dw_{\mathcal{D}} \\ &= -\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x,y,w_{\mathcal{D}})}{p(x,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}}.\end{aligned} \tag{9}$$

Using logarithm arithmetic and adding and subtracting terms we get:

$$\begin{aligned}(\star\star) = &-\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x,y,w_{\mathcal{D}})}{p(x,y)p(w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}} \\ &-\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ p(x,y)p(w_{\mathcal{D}}) \right\} dx\, dy\, dw_{\mathcal{D}} \\ &-\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x)p(w_{\mathcal{D}})}{p(x,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}} \\ &+\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ p(x)p(w_{\mathcal{D}}) \right\} dx\, dy\, dw_{\mathcal{D}}.\end{aligned} \tag{10}$$

Using the law of total probability along with the definitions of the differential entropy and the Lautum information we can reformulate (10) as follows:

$$\begin{aligned}(\star\star) = &L(w_{\mathcal{D}};(x,y)) \\ &+ H(w_{\mathcal{D}}) + H(x,y) \\ &- L(w_{\mathcal{D}};x) \\ &- H(x) - H(w_{\mathcal{D}}).\end{aligned} \tag{11}$$

Since $H(y|x) = H(x,y) - H(x)$ we get that:

$$(\star\star) = L(w_{\mathcal{D}};(x,y)) + H(y|x) - L(w_{\mathcal{D}};x). \tag{12}$$

We next analyze the expression of $(\star)$:

$$(\star) = -\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{f(y|x,w_{\mathcal{D}})}{p(y|x,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}}. \tag{13}$$

Within the $\log$ operation we multiply and divide by the term $\frac{p(x,y)p(w_{\mathcal{D}})}{p(x,w_{\mathcal{D}})}$ and get:

$$\begin{aligned}(\star) = &\iiint p(x,y)p(w_{\mathcal{D}}) \cdot \\ &\log \left\{ \frac{p(x,y)p(w_{\mathcal{D}})}{f(y|x,w_{\mathcal{D}})p(x,w_{\mathcal{D}})} \cdot \frac{p(y|x,w_{\mathcal{D}})p(x,w_{\mathcal{D}})}{p(x,y)p(w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}}.\end{aligned} \tag{14}$$

Since $f(y|x,w_{\mathcal{D}})$ is the learned classifier which outputs the probability of the label y for an input x given the model weights $w_{\mathcal{D}}$, without the labels it has no affect on the joint distribution of the weights and inputs, i.e. $f(x,w_{\mathcal{D}}) = p(x,w_{\mathcal{D}})$, $f(w_{\mathcal{D}}) = p(w_{\mathcal{D}})$. Accordingly,

$$\begin{aligned}(\star) = &\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x,y)p(w_{\mathcal{D}})}{f(x,y,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}} \\ &-\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x,y)p(w_{\mathcal{D}})}{p(x,y,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}}.\end{aligned} \tag{15}$$

Since $f(x,y|w_{\mathcal{D}}) = \frac{f(x,y,w_{\mathcal{D}})}{p(w_{\mathcal{D}})}$ we get that:

$$\begin{aligned}(\star) = &\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x,y)}{f(x,y|w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}} \\ &-\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x,y)p(w_{\mathcal{D}})}{p(x,y,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}}.\end{aligned} \tag{16}$$

In the first term we have the expectation over $w_{\mathcal{D}}$ and so:

$$\begin{aligned}(\star) = &\mathbb{E}_{w_{\mathcal{D}}} \left\{ \iint p(x,y) \log \left\{ \frac{p(x,y)}{f(x,y|w_{\mathcal{D}})} \right\} dx\, dy \right\} \\ &-\iiint p(x,y)p(w_{\mathcal{D}}) \log \left\{ \frac{p(x,y)p(w_{\mathcal{D}})}{p(x,y,w_{\mathcal{D}})} \right\} dx\, dy\, dw_{\mathcal{D}}.\end{aligned} \tag{17}$$

We get that the first term is the expectation over $w_{\mathcal{D}}$ of the KL-divergence between $p(x,y)$ and $f(x,y|w_{\mathcal{D}})$, whereas the second term is the negative Lautum information between (x,y) and $w_{\mathcal{D}}$:

$$(\star) = \mathbb{E}_{w_{\mathcal{D}}} \left\{ KL(p(x,y)||f(x,y|w_{\mathcal{D}})) \right\} - L(w_{\mathcal{D}};(x,y)). \tag{18}$$

Plugging the expressions we got for $(\star\star)$ from (12) and for $(\star)$ from (18) into (8) we obtain the expression in (4):

$$\begin{aligned}(\star) + (\star\star) = &\underbrace{\mathbb{E}_{w_{\mathcal{D}}} \left\{ KL(p(x,y)||f(x,y|w_{\mathcal{D}})) \right\} - L(w_{\mathcal{D}};(x,y))}_{(\star)} \\ &+ \underbrace{L(w_{\mathcal{D}};(x,y)) + H(y|x) - L(w_{\mathcal{D}};x)}_{(\star\star)} \\ = &\mathbb{E}_{w_{\mathcal{D}}} \left\{ KL(p(x,y)||f(x,y|w_{\mathcal{D}})) \right\} + H(y|x) - L(w_{\mathcal{D}};x).\end{aligned} \tag{19}$$

C. Estimating the Lautum information

We are interested in using the Lautum information as a regularization term, which we henceforth refer to as "Lautum regularization". Since computing the Lautum information between two random variables requires knowledge of their probability distribution functions (which are high-dimensional and hard to estimate), we assume that $w_{\mathcal{D}}$ and x are jointly Gaussian with zero-mean. Even though this may seem like an arbitrary assumption, it nevertheless provides ease of computation and good experimental results as shown in Section 4.

Since we only have one instance of the network weights at any specific point during training, we use the network features as a proxy for the network weights in the calculation of the Lautum information, instead of using the weights themselves. Namely, we use the network's output (its pre-softmax logits) when the input is x as a proxy for the network weights $w_{\mathcal{D}}$. This way we have in every training iteration a number of samples equivalent to the size of our training mini-batch, instead of only one sample which would not allow any stable estimation to be made.

As shown in [4], the Lautum information between two jointly Gaussian random variables (w, x) with covariance

$$\begin{bmatrix} \Sigma_w & \Sigma_{wx} \\ \Sigma_{xw} & \Sigma_x \end{bmatrix}, \tag{20}$$

where $\Sigma_x \succ 0$ and $\Sigma_w \succ 0$, is given by

$$\begin{aligned} L(w;x) = &\log\left\{\det(I - \Sigma_x^{-1}\Sigma_{xw}\Sigma_w^{-1}\Sigma_{wx})\right\} \\ &+ 2tr((I - \Sigma_x^{-1}\Sigma_{xw}\Sigma_w^{-1}\Sigma_{wx})^{-1} - I). \end{aligned} \tag{21}$$

The covariance matrix of our target dataset Σ_x is evaluated once before training using the entire target training set, whereas $\Sigma_w, \Sigma_{wx}, \Sigma_{xw}$ are evaluated during training using the current mini-batch in every iteration. All of these matrices are estimated using standard sample covariance estimation based on the current mini-batch in every iteration, e.g.

$$\Sigma_x = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} (x_i - \mu_x)(x_i - \mu_x)^T, \tag{22}$$

and

$$\Sigma_{xw} = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} (x_i - \mu_x)(w_i - \mu_w)^T, \tag{23}$$

where

$$\mu_x = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} x_i, \qquad \mu_w = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} w_i$$

represent the sample mean values of x and w respectively (i.e. their average values in the current mini-batch) and N_{batch} denotes the mini-batch size. The dimensions of these matrices are $\Sigma_x \in \mathbb{R}^{D\times D}, \Sigma_w \in \mathbb{R}^{K\times K}, \Sigma_{xw} \in \mathbb{R}^{D\times K}, \Sigma_{wx} \in \mathbb{R}^{K\times D}$. Note that $\Sigma_{wx} = \Sigma_{xw}^T$ hence only one of these matrices has to be calculated from the samples in every training iteration. Since these are high-dimensional matrices, obtaining a numerically stable sample estimate would require a large amount of data, which would require the use of a very large mini-batch. This constraint poses both a hardware problem (since standard GPUs cannot fit mini-batches of thousands of examples) and a potential generalization degradation (as smaller mini-batches have been linked to improved generalization [1]).

To overcome this issue, we use a standard exponentially decaying moving-average estimation of the three matrices $\Sigma_w, \Sigma_{wx}, \Sigma_{xw}$ in order to obtain numerical stability. We denote by α the decay rate, and get the following update rule for the three covariance matrices in every training iteration:

$$\Sigma_{(n)} = \alpha\Sigma_{(n-1)} + (1-\alpha)\Sigma_{batch}, \tag{24}$$

where n denotes the training iteration and Σ_{batch} denotes the sample covariance matrix calculated using the current mini-batch. Using an exponentially decaying moving-average calculation is of particularly high importance for Σ_w, which is inverted to compute the Lautum regularization term.

D. Experimental setup details

As presented in Section 3.1, our training consists of two stages. First, we train the network using our fully labeled source training set while using Lautum regularization with the unlabeled samples from our target training set. We use the same mini-batch size both for the calculation of the cross-entropy loss (using labeled source samples) and for the computation of the Lautum regularization term (using unlabeled target samples). We also use an exponentially decaying moving average to obtain numerical stability in the estimation of the the covariance matrices $\Sigma_w, \Sigma_{wx}, \Sigma_{xw}$. The matrix Σ_x is calculated once before training and remains constant all throughout it.

Second, we perform a transfer to the target set by training (fine-tuning) the entire network using the labeled samples from the target training set, where the mini-batch size remains the same as before. As in [5], we fine-tune the entire network since this best fits the settings of semi-supervised learning. As mentioned above, we do not apply the Lautum regularization at this stage as we empirically found that it does not improve the results.

We perform our experiments on the MNIST and CIFAR-10 datasets. For MNIST we examine the transfer to the notMNIST dataset, which consists of 10 classes representing the letters A-J. The notMNIST dataset is similar to the MNIST dataset in its grayscale styling and image size, yet it

differs in content. For CIFAR-10 we examine the transfer to 10 specific classes of the CIFAR-100 dataset (specifically, classes 0, 10, 20,...,90 of CIFAR-100, which we reclassified as classes 0, 1, 2,...,9 respectively). These CIFAR-100 classes are different than the corresponding CIFAR-10 ones in content. For example, class 0 in CIFAR-10 represents airplanes whereas class 0 in CIFAR-100 represents beavers etc. Both in the MNIST $\rightarrow$ notMNIST case and the CIFAR-10 $\rightarrow$ CIFAR-100 (10 classes) case we use the same CNN as in [3]. The architecture of the network is illustrated in Appendix E.

D.1. MNIST to notMNIST experimental details

In order for the input images to fit the network's input we resized the MNIST and notMNIST images to 32x32 pixels and transformed each of them to RGB format. Training was done using an Adam optimizer [2] and a mini-batch size of 50 inputs. With this network and using standard supervised training on the entire MNIST dataset we obtained a test accuracy of 99.01% on MNIST.

For each of the four transfer learning methods we examined three different splits of the notMNIST training dataset which consists of 200,000 samples to an unlabeled part and a labeled part: (1) unlabeled part of 199,950 samples and a labeled part of 50 samples; (2) unlabeled part of 199,900 samples and a labeled part of 100 samples; (3) unlabeled part of 199,800 samples and a labeled part of 200 samples. All three options use very few labeled samples in order to fairly represent realistic semi-supervised learning scenarios - in all three options 99.9% or more of the training data is unlabeled. We used a decay rate of $\alpha = 0.999$ for the exponentially decaying moving average estimation of the 3 covariance matrices $\Sigma_w, \Sigma_{wx}, \Sigma_{xw}$, and a different value of λ (which controls the weight of the Lautum regularization) in each scenario which we found to provide a good balance between the cross-entropy loss and the Lautum regularization.

D.2. CIFAR-10 to CIFAR-100 (10 classes) experimental details

In the CIFAR-10 $\rightarrow$ CIFAR-100 (10 classes) case training was done using an Adam optimizer [2] and a mini-batch size of 100 inputs. With this network and using standard supervised training on the entire CIFAR-10 dataset we obtained a test accuracy of 85.09% on CIFAR-10.

Our target set consists of 10 classes of the CIFAR-100 dataset. Accordingly, our training target set consists of 5,000 samples and our test target set consists of 1,000 samples. We examined the same four transfer learning techniques as in the MNIST $\rightarrow$ notMNIST case, where for each we examined three different splits of the CIFAR-100 (10 classes) training dataset to an unlabeled part and a labeled part: (1) unlabeled part of 4,900 samples and a labeled part of 100 samples; (2) unlabeled part of 4,800 samples and a labeled part of 200 samples; (3) unlabeled part of 4,500 samples and a labeled part of 500 samples. All three options use a small number of labeled samples in order to fairly represent realistic semi-supervised learning scenarios - in all three options 90% or more of the data is unlabeled. We used a decay rate of $\alpha = 0.999$ for the exponentially decaying moving average estimation of the 3 covariance matrices $\Sigma_w, \Sigma_{wx}, \Sigma_{xw}$, and a different value of λ (which controls the weight of the Lautum regularization) in each scenario which we found to provide a good balance between the cross-entropy loss and the Lautum regularization.

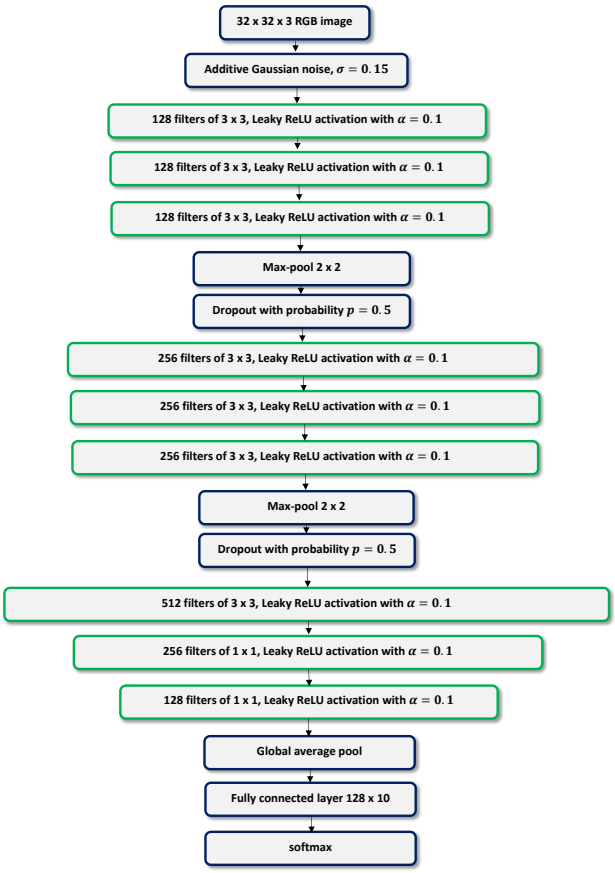

Figure 1: The network architecture used in our experiments.

E. The CNN architecture used in the experiments

Both in the MNIST $\rightarrow$ notMNIST case and the CIFAR-10 $\rightarrow$ CIFAR-100 (10 classes) case we used the same CNN as in [3]. The architecture is illustrated in Figure 1.

References

[1] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[3] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

[4] D. P. Palomar and S. Verdu. Lautum information. *IEEE Trans. Inform. Theory*, 54(3):964–975, March 2008.

[5] Hong-Yu Zhou, Avital Oliver, Jianxin Wu, and Yefeng Zheng. When semi-supervised learning meets transfer learning: Training strategies, models and datasets. *arXiv*, abs/1812.05313, 2018.