Feature-Matching Auto-Encoders

Dustin Tran
Columbia University

Yura Burda
OpenAI

Ilya Sutskever
OpenAI

Abstract

We examine how learning from unaligned data can improve both the data efficiency of supervised tasks as well as enable alignments without any supervision. For example, consider unsupervised machine translation: the input is two corpora of English and French, and the task is to translate from one language to the other but without any pairs of English and French sentences. To address this, we develop feature matching auto-encoders (FMAEs). FMAEs ensure that the marginal distribution of feature layers is preserved across forward and inverse mappings between domains. FMAEs achieve state of the art for semi-supervised neural machine translation with significant BLEU score differences of up to 5.7 and 6.3 over traditional supervised models. Furthermore, on English-to-German, FMAEs outperform last year's best models such as ByteNet [8] while using only half as many supervised examples.

1 Introduction

Massive collections of supervised data have been essential to deep learning advances such as image classification [11], neural machine translation [16], and more recently, cross-domain and intra-domain alignments such as text-to-image synthesis [14] and image-to-image translation [5]. However, perceptual domains most often arise without explicitly aligned pairs. Supervised examples are human-labelled, which presents a fundamental bottleneck in learning from natural images or language.

In this paper, we examine how learning from unaligned data can improve both the data efficiency of supervised tasks as well as enable alignments without any supervision. For example, consider unsupervised machine translation: the input is simply two corpora of English and French, and the task is to translate from one language to the other but without any pairs of English and French sentences. More generally in text, tasks often involve taking a source sentence as input and returning a target sentence with a shared representation as the input but with target-specific properties; other examples include text decipherment and "style transfer" of sentiment, authors, and/or genres.

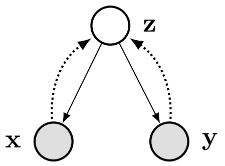
In this work, we develop *feature matching auto-encoder*s *(*FMAEs*)*. We show empirically that FMAEs learn powerful and alignments for neural machine translation. Most compellingly, FMAEs achieve state of the art for semi-supervised neural machine translation with significant BLEU score differences of up to 5.7 and 6.3 over traditional supervised models. Furthermore, on English-to-German, FMAEs outperform last year's best models such as ByteNet [8] while using only half as many supervised examples.

2 Latent Variable Model for Alignment

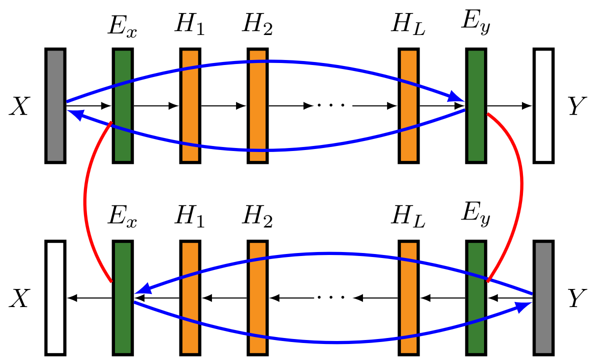
We formalize the problem and then derive a principled approach with probabilistic models. There are two data sets of i.i.d observations $\{x\} \sim p_{\text{data}}(x)$ and $\{y\} \sim p_{\text{data}}(y)$. Each data point is a variable-length sequence of discrete values, $x = (x_1, \dots, x_T)$, and similarly for y.[1] The goal is to

[1]For now, we assume there are no supervised examples, which are pairs $\{(x, y)\} \sim p_{\text{data}}(x, y)$.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.



(a) Latent variable model where z represents invariant structure across domains. Dotted lines represent inference. With unaligned data, we only observe individual x's and y's and not pairs.



(b) Feature matching auto-encoders (FMAEs). Each alignment mapping (top and bottom) is written as a composition of feature maps with embedding layers E_x, E_y and a sequence of hidden layers H_ℓ. FMAEs minimize reconstruction error (blue) plus an adversarial penalty which matches the marginal distributions over feature layers (red); in experiments, we only match embedding layers.

learn a mapping between the domains $G : X \to Y$, or conversely, $F : Y \to X$. Given a test input in one domain, this lets us predict the output in the other.

With probabilistic generative models, a natural approach is to posit a generative process according to the factorization $p(z)p(x \mid z)p(y \mid z)$ (Figure 1a). The latent variable z has a fixed prior, and each domain's observations are drawn conditionally independent given z via a neural network. This model has been studied as a principle for one shot learning and domain adaptation [15], and has also been revisited for multimodal learning [17, 20, 4, 19].

Using maximum likelihood, we maximize the following with respect to model parameters,

$$\mathbb{E}_{p_{\mathrm{data}}(x)}[\log p(x)] + \mathbb{E}_{p_{\mathrm{data}}(y)}[\log p(y)].$$

Maximizing is equivalent to minimizing the negative marginal density. Standard variational methods posit an upper bound on this loss [7],

$$\begin{aligned} &\mathbb{E}_{p_{\mathrm{data}}(x)q(z \mid x)}[-\log p(x \mid z)] + \mathbb{E}_{p_{\mathrm{data}}(y)q(z \mid y)}[-\log p(y \mid z)] + \\ &\mathbb{E}_{p_{\mathrm{data}}(x)}[\mathrm{KL}(q(z \mid x) \,\|\, p(z))] + \mathbb{E}_{p_{\mathrm{data}}(y)}[\mathrm{KL}(q(z \mid y) \,\|\, p(z))]. \end{aligned} \tag{1}$$

The first two terms can be interpreted as "reconstruction errors", which determine the average number of bits to capture a data point x (y) under noisy encodings. The last two terms are divergences which regularize the individual encoders; it shares information across domains via shrinkage toward the prior. After training, the model performs alignments by composing encoders with decoders: for an input x, map $x \mapsto z$ via $q(z \mid x)$ and $z \mapsto y$ via $p(y \mid z)$; the converse holds for an input y.

Empirically (§ 4), we found that a latent variable model trained with Equation 1 fails to learn alignments. From a statistical point of view, selecting the right prior is key to successfully share across domains, and it is difficult to specify our assumptions about alignment this way. From a computational point of view, difficulties exist in optimizing the variational objective while utilizing the latent code. Using this model as motivation, we design a method overcoming these problems.

3 Feature Matching Auto-Encoder

The latent variable model in § 2 describes alignment as a composition of encoder-to-decoders $X \to Z \to Y$ and $Y \to Z \to X$. Here, we consider alignment mappings under finer granularity as a sequence of feature maps. Figure 1b displays a composition of two embedding layers and L hidden layers for each alignment mapping $G : X \to Y$ and $F : Y \to X$. For layer indices $\ell = 0, \dots, L+1$, denote individual feature maps as $g_\ell : H_{\ell-1} \to H_\ell$ for G and $f_\ell : H_\ell \to H_{\ell-1}$ for F; $\ell = 0$ and $L + 1$ include the embeddings as domain and range respectively.

For a layer $\ell \in \{0, \dots, L + 1\}$, consider the invariance property

$$p(g_\ell(x)) = p(f_{L+1-\ell}(y)). \tag{2}$$

Equation 2 says that the marginal distribution of a feature layer should be the same regardless of whether the layer is induced by the data distribution on x (left hand side) or if the layer is induced by the data distribution on y (right hand side).

If the number of hidden layers $L = 1$, this invariance reduces to matching the distribution of the middle layer in the mappings $X \to Z \to Y$ and $Y \to Z \to X$. This mimicks the distribution invariance in § 2 where two KL divergences penalize deviations from a fixed prior distribution; however, Equation 2 posits a single divergence which penalizes deviation from each other. This pulls information across domains with an implicit, learnable density on the features. It indirectly posits a prior over the shared space without the need to specify a fixed, tractable prior density.

Unlike the latent variable model, feature matching also matches across arbitrary layers in a neural network. This lets us perform matching on feature layers closest to data space (namely, the embedding layers) while still avoiding the difficulties of adversarial training directly on discrete sequences. Note this also avoids the issue of latent code utilization in the decoder: matching embedding layers forces the decoder to use the encoder output in order to marginally match the distribution of the outputted embedding layer.

To enforce Equation 2, we apply an adversarial penalty jointly over the desired feature layers. Namely, to match the two embedding layers, consider the penalty

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})p_{\text{data}}(\mathbf{y})}[f(\mathbf{e}_x, \mathbf{e}_y)] - \mathbb{E}_{p_{\text{data}}(\mathbf{x})p_{\text{data}}(\mathbf{y})}[f(\mathbf{e}_x, \mathbf{e}_y)], \tag{3}$$

where in the first term, $\mathbf{e}_x$ is set via the input embedding layer $X \to E_x$ and $\mathbf{e}_y$ is set to the output embedding via $X \to E_x \to \cdots \to E_y$; in the second term, the converse holds where $\mathbf{e}_x$ is set to the output embedding via $Y \to E_y \to \cdots \to E_x$ and $\mathbf{e}_y$ is set via the input embedding layer $X \to E_x$. With a discriminator over 1-Lipschitz functions, the max over f is equal to a Wasserstein distance between the marginal embedding distributions [9].

Using the penalty of Equation 3, FMAEs minimize an objective with reconstruction terms,

$$\begin{aligned}&\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[-\log p(\mathbf{x} \mid \mathbf{e}_y)] + \mathbb{E}_{p_{\text{data}}(\mathbf{y})}[-\log p(\mathbf{y} \mid \mathbf{e}_x)] + \\ &\lambda\Big(\mathbb{E}_{p_{\text{data}}(\mathbf{x})p_{\text{data}}(\mathbf{y})}[f(\mathbf{e}_x, \mathbf{e}_y)] - \mathbb{E}_{p_{\text{data}}(\mathbf{x})p_{\text{data}}(\mathbf{y})}[f(\mathbf{e}_x, \mathbf{e}_y)]\Big).\end{aligned} \tag{4}$$

Using Monte Carlo samples and reparameterization gradients [10], we minimize Equation 4 with respect to encoder-decoder parameters and we maximize it with respect to the discriminator f. Both enable backpropagation.

Semi-Supervised Learning. FMAEs extend to learn from supervised examples in addition to unaligned (unsupervised) ones. Namely, to learn from paired examples $\{(x, y)\} \sim p_{\text{data}}(x, y)$, we add the typical likelihood terms to Equation 4 to encourage $G(x) = y$ and $F(y) = x$ per data point,

$$\lambda_{\text{sup}}\, \mathbb{E}_{p_{\text{data}}(\mathbf{x},\mathbf{y})}[-\log p(\mathbf{x} \mid \mathbf{y}) - \log p(\mathbf{y} \mid \mathbf{x})],$$

where $\lambda_{\text{sup}} \in \mathbb{R}^+$ balances how much we weigh aligned examples over unaligned examples.

4 Experiments: Neural Machine Translation

Table 1 displays results using FMAEs as well as current state-of-the-art translation models. For comparison on limited supervision, we also trained the Transformer network (with same modified architecture as the FMAE's); it only uses the available supervised examples. In one version we used the same architecture and hyperparameters as one mapping in the FMAE (thus it has half the total number of parameters); in another version, we doubled the attention layer sizes to have comparable size in its single mapping to FMAE's dual mappings. The FMAE significantly outperforms the Transformer network. The BLEU scores have a dramatic difference from 2.5 and 5.4 on 2M supervised examples to up to 5.7 and 6.3 on 500K supervised examples.

Given the equivalent amount of supervision, the unaligned data set size in EN-FR (roughly 34-35M sentences per corpus) enables the FMAE to have improved BLEU scores in EN-FR over EN-DE. Most compellingly, we also note that our results for EN-DE outperformed last year's results of ByteNet [8] while using only half as many supervised examples.

References

[1] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. In *International Conference on Machine Learning*.

[2] Frey, B. J. (1997). Continuous sigmoidal belief networks trained using slice sampling. In *Neural Information Processing Systems*.

[3] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

[4] Higgins, I., Sonnerat, N., Matthey, L., Pal, A., Burgess, C. P., Botvinick, M., Hassabis, D., and Lerchner, A. (2017). SCAN: Learning Abstract Hierarchical Compositional Visual Concepts. *arXiv preprint arXiv:1707.03389*.

[5] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. In *Conference on Computer Vision and Pattern Recognition*.

[6] Jang, E., Gu, S., and Poole, B. (2017). Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[7] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An Introduction to Variational Methods for Graphical Models. *Machine Learning*.

[8] Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

[9] Kantorovich, L. V. and Rubinstein, G. S. (1958). On a space of completely additive functions. *Vestnik Leningrad. Univ*, 13(7):52–59.

[10] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

[11] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems*, pages 1097–1105.

[12] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. In *Empirical Methods on Natural Language Processing*.

[13] Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The Concrete Distribution: A Concrete Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*.

[14] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016). Generative Adversarial Text to Image Synthesis. In *International Conference on Machine Learning*.

[15] Sutskever, I., Jozefowicz, R., Gregor, K., Rezende, D. J., Lillicrap, T., and Vinyals, O. (2015). Towards principled unsupervised learning. *arXiv preprint arXiv:1511.06440*.

[16] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In *Neural Information Processing Systems*.

[17] Suzuki, M., Nakayama, K., and Matsuo, Y. (2016). Joint Multimodal Learning with Deep Generative Models. *arXiv preprint arXiv:1611.01891*.

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Neural Information Processing Systems*.

[19] Vedantam, R., Fischer, I., Huang, J., and Murphy, K. (2017). Generative Models of Visually Grounded Imagination. *arXiv preprint arXiv:1705.10762*.

[20] Wang, W., Yan, X., Lee, H., and Livescu, K. (2016). Deep Variational Canonical Correlation Analysis. *arXiv preprint arXiv:1610.03454*.

[21] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016).

Method on EN-DE	# of Supervised Examples (BLEU)			
	500K	1M	2M	4.5M
Transformer ($\approx$65M params)	12.3	14.6	20.3	
Transformer ($\approx$130M params)	10.3	18.5	21.5	
Feature matching auto-encoder	**16.0**	**21.4**	**24.0**	
Transformer (“big”) [18]				**28.4**
Conv Seq2Seq [3]				25.16
Google NMT [21]				24.6
ByteNet [8]				23.75
RNN Enc-Dec-Att [12]				20.9
RNN Enc-Dec [12]				14.0

Method on EN-FR	# of Supervised Examples (BLEU)			
	500K	1M	2M	36M
Transformer ($\approx$65M params)	12.1	15.1	18.9	
Transformer ($\approx$130M params)	10.8	14.6	21.3	
Feature matching auto-encoder	**17.1**	**23.1**	**26.7**	
Transformer (“big”) [18]				**41.0**
Conv Seq2Seq[3]				40.46
Google NMT [21]				39.92

Table 1: (top) BLEU scores on EN-DE newstest2014 test set while trained over a fixed number of supervised examples. **(bottom)** BLEU scores on EN-FR newstest2014 test set while trained over a fixed number of supervised examples. FMAEs outperform existing methods for semi-supervised translation with significant BLEU score differences.

Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

[22] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*.

A Experiment Details

In all experiments, we used the Adam optimizer with an initial step-size of one of $\{0.001, 0.0005, 0.0001\}$; we set $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-8}$. For machine translation, we followed the learning rate schedule of Vaswani et al. [18] which increases the learning rate linearly for a fixed number of warm-up steps during training followed by a decrease proportional to the inverse square root of the step number. For text decipherment and sentiment transfer, we used a batch size of 256 unaligned input and output sequences. Sequences were batched together by approximate sequence length. Convolutional filters and weight matrices were initialized with Glorot uniform; embeddings initialized uniformly between $[-0.1, 0.1]$; biases initialized at 0.

For machine translation, we used the standard data sets of WMT 2014 EN-DE and WMT 2014 EN-FR. The WMT 2014 EN-DE data set consists of roughly 4.5 million sentence pairs. Following Vaswani et al. [18], we encoded sentences using byte-pair encoding, producing a shared source-target vocabulary of about 37,000 tokens. The WMT 2014 EN-FR data set consists of a much larger corpus of 36 million sentence pairs. We split tokens into a 32,000 word-piece vocabulary [21]. To simulate a semi-supervised task, we partitioned both data sets into a fixed number of supervised pairs and made the rest unaligned. In both tasks, we evaluate performance with newstest2014 as test set and newstest2013 as validation set.

A.1 Architecture: Attention

For the network architectures, we primarily follow the Transformer of Vaswani et al. [18], which has seen strong success for sequence-to-sequence modeling. The Transformer uses only attention layers for both the encoder and decoder; see **??**. The encoder applies an embedding layer to inputs followed by L layers of self-attention. The decoder applies an embedding layer followed by L alternations of a self-attention layer and a layer which attends over the encoder hidden states.

We make three adjustments which we found improved our experiments. First, we add noise to attention layers in order to sparsify the locations to attend to; we provide detail in the next subsection (Appendix A.2). Second, we apply layer norm to the input of each residual block instead of afterwards. Third, we use learnable positional encodings rather than fixed sinusoidal embeddings as a way to impose ordering in the sequences [3]. Namely, the embedding layers take input elements $x = (x_1, \dots, x_T)$, perform a table look-up to obtain its word embedding free parameters $w = (w_1, \dots, w_T)$, and sums it with positional free parameters $(p_1, \dots, p_T)$ to return the embedding, $e_x = (w_1 + p_1, \dots, w_T + p_T)$.

For the discriminator, we also apply L self-attention layers and clip weights following Arjovsky et al. [1]. As the matching distributions assume independence among features, we parameterize the discriminator to not include interactions among its inputs. A problem when training Equation 4 is that we are matching the discriminator across two distributions with free parameters. This produces difficulties because the mappings can constantly scale the output of f so long as the relative difference remains the same. To address this, we simply L2-normalize the inputs to f.

A.2 Adding Noise to Attention

An attention function can be described as taking a query and memory as input and returning a weighted sum over the memory states. The query and each memory state are vectors such as a decoder hidden state and the set of encoder hidden states respectively. This is also known as "soft attention," which is equivalent to taking an expectation over a categorical variable $z \in \{1, \dots, T\}$ which attends to a specific memory state,

$$\mathbb{E}_{p(z \mid \mathbf{M}, q)}[\mathbf{M}_z] = \sum_{t=1}^{T} \pi_t \mathbf{M}_t,$$

where z's distribution is a function of the matrix of memory states $\mathbf{M}$ and query vector q.

Soft attention produces dense weights where all states have a nonzero probability. Many tasks only require attending over few inputs such as machine translation, which often only requires finding the corresponding word to translate and its context. In order to sparsify the attended locations, we add noise to the softmax inputs: given a T-dimensional vector of logits inputs, return

$$\pi = \mathrm{softmax}((\mathrm{logits} + g)/\tau), \quad g = (g_1, \dots, g_T), g_t \sim \mathrm{Gumbel}(0, 1).$$

We use $\tau = 0.1$ in experiments. This forces the inputs to robustify against noise by taking on large positive or negative values [2]. It is equivalent to a sample from the Gumbel-Softmax distribution and admits backpropagation [6, 13]. It can be interpreted as a relaxation of "hard attention" [22], which requires score function gradients to handle the discrete variable. Adding noise augments each attention layer as a stochastic layer in a deep latent variable model; the temperature parameter bridges from hard to soft attention.[2]

[2]Empirically we find setting the temperature τ arbitrary close to 0 is undesirable. The model benefits from attending to few but multiple locations, whereas hard attention assumes only one.